MICROCOMPUTADOR CURSO PRÁTICO

TIRE O MÁXIMO DO MICRO EM 30 SEMANAS

PUBLICAÇÃO SEMANAL ILUSTRADA

Cr$ 12.900

30

Visão do futuro • Divisão binária
Aprendizagem evolutiva

COM ESTA EDIÇÃO TERMINA O SEGUNDO E ÚLTIMO VOLUME DE **MICROCOMPUTADOR CURSO PRÁTICO**
ADQUIRA A CAPA DURA PARA ENCADERNÁ-LO.

rioGráfica

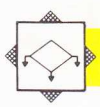
MANAUS, ALTAMIRA, MACAPÁ, JI-PARANÁ, PORTO VELHO, RIO BRANCO, SANTARÉM, BOA VISTA, SINOP, ALTA FLORESTA: Cr$ 16.770

MICROCOMPUTADOR CURSO PRÁTICO

APLICAÇÕES

Aprendendo a aprender 597

Visão do futuro 605



LINGUAGENS

Divisor de águas 601

SUMÁRIO 607

ÍNDICE GERAL 609

PLANO DA OBRA

MICROCOMPUTADOR - CURSO PRÁTICO é uma coleção de 30 fascículos de periodicidade semanal. Os 29 primeiros terão, cada um, vinte páginas internas (miolo) e quatro capas, além de uma introdução de dezesseis páginas acompanhando o fascículo 1. O fascículo 30 incluirá o índice geral da obra. Os miolos, encadernados, constituem dois volumes de um curso prático de microcomputação, com seções indicadas por tarjas de diferentes cores.
As duas primeiras capas são descartáveis; as 3.as e 4.as trazem programas de jogos. Encadernadas, formam um volume de cem páginas.

COMO ENCADERNAR

As capas duras, incluindo as guardas, estarão à venda simultaneamente com os fascículos 10 (**Volume 1**) e 30 (**Volume 2** e **Jogos**).
Volume 1 - Deve ser encadernado com os elementos dispostos nesta ordem: guardas; frontispício e introdução (publicados com o fascículo 1); miolos dos fascículos de 1 a 15; e guardas.
Volume 2 - Guardas; frontispício (que virá com o fascículo 30); miolos dos fascículos de 16 a 30; e guardas.
Jogos - Guardas; frontispício e sumário (que constituem as 3.as e 4.as capas do fascículo 30); 3.as e 4.as capas dos fascículos de 1 a 29; e guardas.

rioGráfica **CONSELHO DE ADMINISTRAÇÃO:** Roberto Irineu Marinho; João Roberto Marinho; José Roberto Marinho; Oscar Neves; **Diretoria:** Oscar Neves; Danilo Esteves Costa; João J. Noro; Marcos March; Nilo Sérgio de Almeida; **GRUPO EDUCAÇÃO E CULTURA - Divisão Comercial - Diretor:** Jaime Rodrigues; **Gerente de Planejamento:** João de Azeredo Coutinho; **Gerente de Vendas:** Rubens Barbosa; **Coordenadores de Promoção:** Paulo Cesar M. Seixas; Edgard Mello M. Neto; **Gerente de Circulação:** Norberto Martin; **Supervisor de Operações:** Abel Pereira Pinto; **Editorial - Diretor:** João J. Noro; **Editor-chefe:** Mauricio Rittner; **Diretor de Arte:** Adalberto Cornavaca; **Administração Editorial:** Heitor de Souza Paixão; **Editor:** Jesse Navarro; **Editor-assistente:** Claudio Marcondes; **Coordenadora de Revisão:** Isis Augusta Loyolla; **Revisores:** Cacilda Guerra; Carlos Eduardo Russo Tranjan; Maria da Graça Mendonça Couto; Sérgio Roberto Torres; **Chefe de Arte:** Bonifácio Duardes Miranda; **Chefe de Estúdio:** José Yuji Kuribayashi; **Assistentes de Arte:** Miguel Luis Escámez Simón; Edemilson Ribeiro da Silva; Wagner Celestino; **Colaboradores - Editores-assistentes:** Isa Mara Lando; Luiz Carlos Pizarro Marin; **Texto:** Carlos Eduardo Silveira Matos; José Antônio Arantes; Jussara Maturo; Paulo Brito; Solange A. Menezes; **Preparação de Texto:** Luiz Carlos Cardoso; **Tradução:** Luiz Roberto de Godoy Vidal; Maria Clara Cescato; Maria Ester Menezes Martinho; Fernando Dorfman Knijnik; Newton Roberval Eichemberg; **Pesquisa:** Stella Maria Quentel; **Assessoria Técnica:** Geraldo Coen; Gustavo José Ferreira Grubba; José Rubens Salles Toledo; Lauro de Lauro; Pierluigi Piazzi; **Estúdio:** Vagner Facuri de Oliveira. As fotos não creditadas pertencem à obra original. © APSIF Copenhagen, 1984; © Orbis Publishing Ltd., 1984; © Editora Rio Gráfica Ltda., 1985, para a língua portuguesa. **Composição:** Editora Rio Gráfica Ltda. e AM Produções Gráficas Ltda. **Impressão:** JBIG.

NÚMEROS ATRASADOS - Você poderá comprar os exemplares que faltam em sua coleção, pelo preço do último fascículo posto à venda, em sua banca de jornal preferida. Caso não consiga obtê-los, faça sua solicitação por carta endereçada diretamente à Editora Rio Gráfica Ltda., rua Itapiru, 1209, Rio Comprido, Rio de Janeiro, CEP 20251. O atendimento será feito por via postal. Nas cidades do Rio de Janeiro e de São Paulo, as compras poderão ser efetuadas pessoalmente nos seguintes endereços: Rio de Janeiro - Rua Itapiru, 1209, Rio Comprido; São Paulo - Rua Frei Caneca, 1152, Consolação.

Distribuidor exclusivo para todo o Brasil: Fernando Chinaglia - Rua Teodoro da Silva, 907, Rio de Janeiro. Distribuidor para Portugal: Electroliber Ltda. - Rua Prof. Reinaldo dos Santos, 1488, Lisboa.

APRENDENDO A APRENDER

Os pesquisadores da inteligência artificial procuram desenvolver "sistemas-criança", que aprendem por si mesmos. Um exemplo é o BEAGLE, um procriador de regras baseado na teoria da seleção natural.

Se um sistema de computador melhorou seu desempenho numa tarefa, durante um certo tempo, sem ser reprogramado, é justo dizer que aprendeu. Isso implica a presença de um padrão para o sistema. Sem uma medida comparativa, não se pode falar em aprendizado.

Um algoritmo de aprendizagem procura realizar um ou mais dos seguintes objetivos:

- Cobrir uma faixa maior de problemas;
- Dar soluções mais precisas;
- Obter respostas com um mínimo de dispêndio; e
- Simplificar o conhecimento codificado.

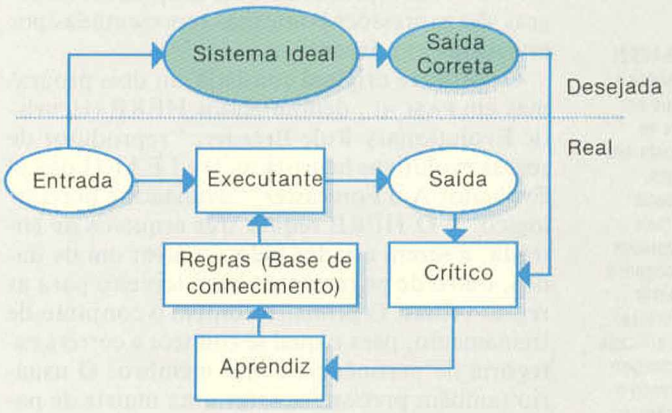
Os sistemas de aprendizagem mais bem-sucedidos têm sido aplicados a problemas de classificação, nos quais se devem examinar os dados de entrada e, de algum modo, classificá-los e identificá-los ou interpretá-los. Além disso, os métodos em que estamos interessados devem ser capazes de generalizar, oferecendo uma resposta apropriada a uma nova situação.

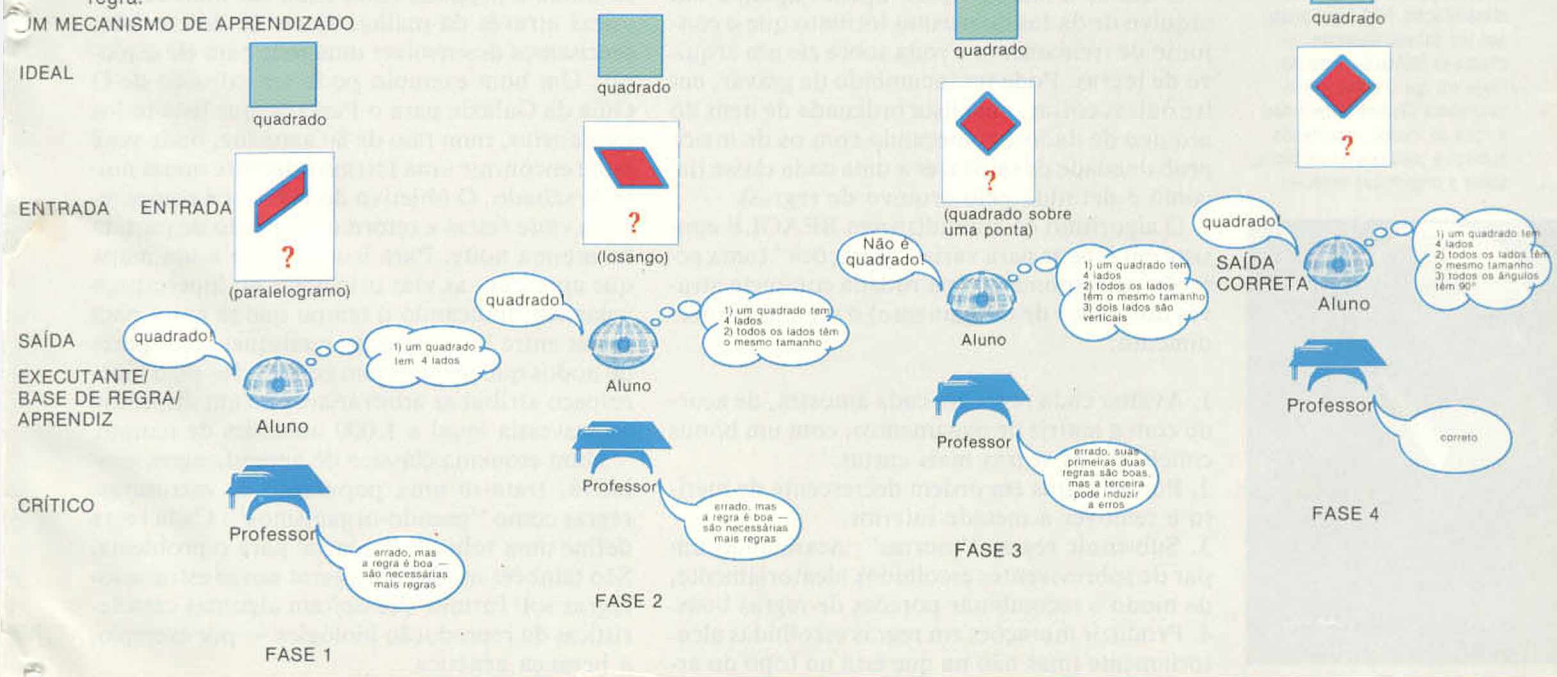
Qualquer sistema de aprendizado de máquina deve incluir os seguintes componentes:

- Um conjunto de estruturas de dados, que representa o nível de habilidade do sistema num determinado momento (as regras);

Intelecto crítico

A maneira pela qual uma criança aprende é complexa e só parcialmente compreendida. Alguns psicólogos sugerem que elas aprendem compondo "esquemas" — estruturas de regras —, muitas vezes por um processo de tentativa e erro: testando hipóteses (as regras) e conservando as que dão resultados corretos. Em muitas situações, o processo de aprendizagem é orientado pelo crítico (o professor), que ajuda a criança (o aprendiz) a avaliar e refinar o conjunto interno de regras que irão ajudá-la a melhorar seu desempenho numa tarefa. Os sistemas de aprendizagem tentam reproduzir essa assistência, criando uma base de conhecimento que pode ser utilizada juntamente com exemplos de treinamento e um método de avaliar o desempenho de cada regra.





- Um algoritmo de tarefa (o executante), que utiliza as regras para guiar a atividade de resolução de problemas;
- Um módulo de retroalimentação (o crítico), que compara resultados obtidos e metas propostas;
- O próprio mecanismo de aprendizagem (o aprendiz), que utiliza a retroalimentação do crítico para aperfeiçoar as regras.

Mas como aperfeiçoá-las? O BEAGLE (Biological Evolutionary Algorithm Generating Logical Expressions, "algoritmo biológico evolutivo gerando expressões lógicas") é um sistema para computador que produz regras de decisão por indução, a partir de um banco de dados. *Beagle* era o nome do navio em que Charles Darwin realizou a viagem de circunavegação do globo terrestre que lhe deu subsídios para a teoria sobre a evolução das espécies — e o sistema baseia-se no princípio darwiniano da seleção natural. As regras que não concordam com os dados vão sendo eliminadas e substituídas por "mutações" melhores ou por outras criadas pelo "acasalamento" com regras mais bem adaptadas. As regras são expressões booleanas representadas por estruturas em árvore.

O software original consistia em dois programas em PASCAL, denominados HERB (Heuristic Evolutionary Rule Breeder, "reprodutor de regras evolutivas heurísticas") e LEAF (Logical Evaluator Aid Forecaster, "avaliador e previsor lógico"). O HERB requer três arquivos de entrada, a serem criados pelo usuário: um de dados, outro de pagamentos e um terceiro para as regras velhas. O primeiro contém o conjunto de treinamento, para o qual se conhece a correta categoria de pertinência como membro. O usuário também precisa fornecer uma matriz de pagamentos, que define o valor ou custo das classificações corretas e incorretas.

O LEAF é mais simples: apenas apanha um arquivo de dados do mesmo formato que o conjunto de treinamento e roda sobre ele um arquivo de regras. Pode ser incumbido de gravar, entre outras coisas, uma lista ordenada de itens do arquivo de dados, começando com os de maior probabilidade de satisfazer a uma dada classe (tal como é definida pelo arquivo de regras).

O algoritmo de aprendizagem BEAGLE consiste em repetir para várias "gerações" (uma geração corresponde a uma rodada completa através dos dados de treinamento) o seguinte procedimento:

1. Avaliar cada regra em cada amostra, de acordo com a matriz de pagamentos, com um bônus concedido às regras mais curtas.
2. Pôr as regras em ordem decrescente de mérito e remover a metade inferior.
3. Substituir regras "mortas", acasalando um par de sobreviventes escolhidos aleatoriamente, de modo a recombinar porções de regras boas.
4. Produzir mutações em regras escolhidas aleatoriamente (mas não na que está no topo do arquivo) e aplicar um procedimento TIDY às novas regras — prontas para a geração seguinte.

O procedimento TIDY elimina duplas negações e outras redundâncias sintáticas eventualmente geradas, "podando" a árvore de regras.

O resultado é que apenas as regras mais bem adaptadas sobrevivem para produzir a geração seguinte, e o arquivo de regras "evolui" por sucessivas gerações — rodadas de treinamento.

Aprendizagem evolutiva

Vejamos, agora, o que acontece quando colocamos em prática uma abordagem darwiniana da aprendizagem de máquina.

Vamos examinar uma versão do famoso PCV (Problema do Caixeiro Viajante). Ele possui uma tabela com a distância entre as 48 capitais de Estados da área continental dos Estados Unidos (excluídos o Alasca e o Havaí); deve visitar cada uma e voltar ao ponto de partida. O objetivo consiste em minimizar a distância total percorrida.

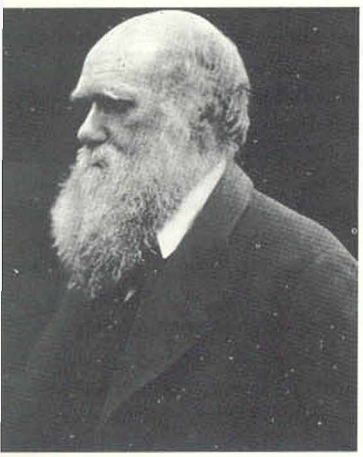
Parece desconcertante de tão simples, mas o número de percursos é de N − 1, sendo N o número de cidades. Para 48 cidades, existem 47 primeiros passos possíveis, seguidos por 46 segundos passos possíveis etc. Na verdade, há mais rotas potenciais do que átomos no universo!

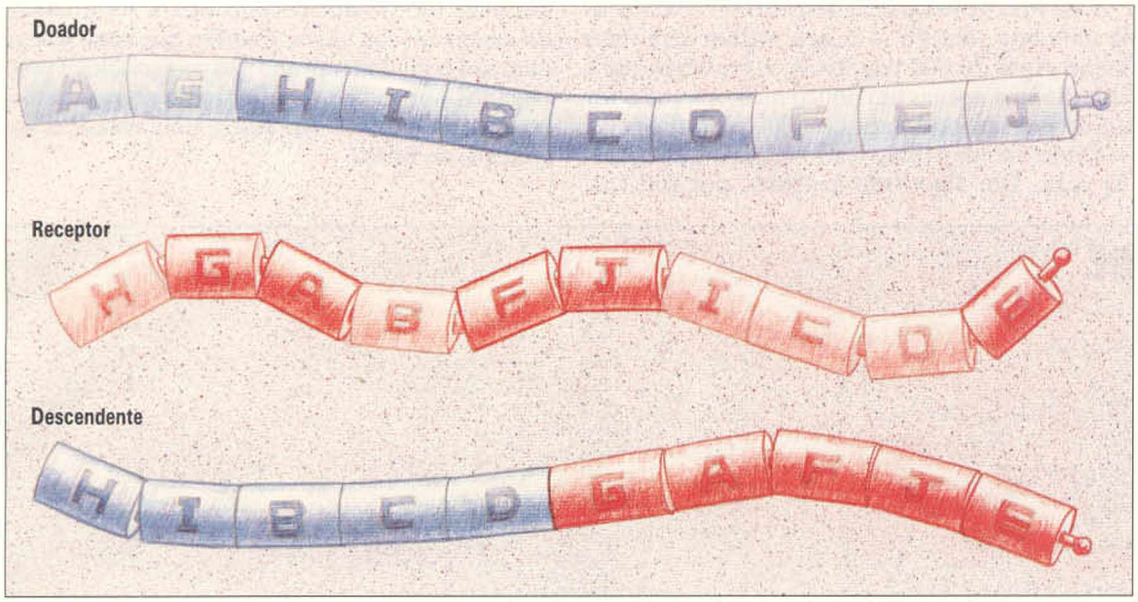
O PCV pode ser tratado como um problema de busca. Também se pode manipulá-lo pelo chamado método de Monte Carlo, que utiliza tentativa e erro aleatoriamente. Entretanto, vamos abordá-lo como um problema de aprendizagem. Não esperamos resgatar a solução ótima, porém uma boa solução.

O programa GENE (General Evolutionary Network Explorer, "explorador geral de rede em evolução"), aqui apresentado, é um sistema de aprendizagem especialmente adaptado para explorar redes e desenvolver regras que evoluem de modo a produzir rotas cada vez mais econômicas através da malha. Antes de descrevê-lo, precisamos desenvolver uma rede para ele explorar. Um bom exemplo pode ser extraído de O Guia da Galáxia para o Penetra, que lista todos os planetas, num raio de 80 anos-luz, onde você pode encontrar uma festinha decente numa noite de sábado. O objetivo do jogador é comparecer a vinte festas e retornar ao ponto de partida na mesma noite. Para isso, recorre a um mapa que apresenta as vias principais no hiperespaço galáctico, indicando o tempo que se gasta para viajar entre duas paradas quaisquer. Aos pares de nodos que não estejam conectados pelo hiperespaço atribui-se arbitrariamente um dispêndio de travessia igual a 1.000 unidades de tempo.

Num esquema clássico de aprendizagem evolutiva, trata-se uma população de estruturas-regras como "pseudo-organismos". Cada regra define uma solução potencial para o problema. São também usadas para gerar novas estruturas-regras sob formas que imitam algumas características da reprodução biológica — por exemplo, a herança genética.

Charles Darwin (1809-1882)
As teorias de Darwin sobre a evolução sustentam que as espécies se adaptam e se aperfeiçoam, em resposta aos ambientes em que vivem, através da seleção natural; apenas os indivíduos mais fortes e mais bem adaptados sobrevivem, para dar origem à geração seguinte e passar adiante suas características. Essa idéia vem sendo aplicada com sucesso à aprendizagem das máquinas, melhorando o desempenho da inteligência artificial nos problemas de ordenamento; desenvolve, para isso, conjuntos de regras de classificação. Não é por acaso que um desses sistemas chama-se BEAGLE, nome do navio em que o então jovem naturalista Charles Darwin deu a volta ao mundo, recolhendo subsídios para sua futura teoria sobre a origem das espécies.





Uma boa estirpe
O GENE cria novos caminhos na rede acasalando duas rotas bem-sucedidas. Como estas correspondem a strings de vinte caracteres, escritos em BASIC, o processo de produzir descendência é essencialmente uma simples manipulação de strings. Neste exemplo, o substring HIBCD é extraído do pai doador e emendado ao pai receptor, assegurando-se de que nenhuma letra seja duplicada. O processo garante, à semelhança do que ocorre nos cruzamentos genéticos reais, a transmissão das características dos pais aos seus descendentes.

Dependendo de seu desempenho, selecionam-se as regras suscetíveis de sobreviver por mais tempo e com maior probabilidade de procriação. No GENE, as regras conservam a forma de strings no programa em BASIC. Na cadeia

ABJHMNCDKTSFRQEGILOP

cada letra denota um planeta na rede e ocorre uma só vez no string. Isto é, um string de vinte caracteres define um percurso particular pela rede. Somando as distâncias para se chegar aos planetas na ordem indicada, podemos avaliar cada string. Quanto menor a combinação das distâncias, melhor o percurso.

Convencionamos que os strings piores que a média sejam apagados após cada "geração". Como substituí-los? O mecanismo de "reprodução sexual" não é o único meio de gerar novos strings: cerca de 8% dos sobreviventes são "mutantes".

A sub-rotina das mutações, que começa na linha 3.500, apenas executa um certo número de permutações aleatórias. A mutação é um operador genético que age em segundo plano, impedindo que o sistema fique preso a uma solução boa, mas passível de melhoria. O padrão médio da população de regras realmente melhora com o tempo, embora isso não seja uma progressão contínua (até as melhores regras podem "morrer").

Nesta altura, colocam-se duas questões interligadas. Até que ponto o método é bom? Por que ele funciona? Para respondê-las, devemos compará-lo a uma típica abordagem Monte Carlo — os algoritmos genéticos são métodos de Monte Carlo modificados. Com esse procedimento, são geradas soluções aleatórias e conservada a melhor solução, dentro de um prazo especificado. Neste exemplo, seria necessário testar uma regra-string aleatória, avaliá-la, conservá-la se for a melhor e fazê-la sofrer mutação. O processo pode ser repetido à vontade.

Engenharia alfabética

Em primeiro lugar, duas estruturas parentais são escolhidas aleatoriamente entre as que sobreviveram ao processo de seleção. Uma delas é denominada "doador" (R1%) e a outra (R2%), "receptor" (ver as linhas 3.030 e 3.050). Extrai-se aleatoriamente uma lasca de "material genético" (um substring) do doador e coloca-se o string S$. Chama-se então a sub-rotina 3.300, que emenda a lasca do doador no receptor, deslocando todos os caracteres do string receptor (exceto os que provieram do doador) na ordem em que aparecem. O processo de acasalamento é, portanto, assimétrico. A união de X com Y não dá o mesmo resultado que a de Y com X, mesmo que as posições aleatórias P1% e P2% sejam idênticas. Vamos examinar um exemplo em escala reduzida (o string "real" tem vinte caracteres). Nos pais

Doador A G H I B C D F E J
Receptor H G A B F J I C D E

podemos selecionar o substring

H I B C D

como sendo a contribuição do doador e uni-lo às letras restantes do receptor, para produzir

H I B C D G A F J E

Deve-se salientar que esse não é o único modo de cruzar um par de regras-strings. Você pode imaginar outros, mas deve garantir — tal como ocorre nos cruzamentos genéticos reais — que segmentos de informação parental sejam transferidos à geração seguinte. Também deve garantir que cada acasalamento produza uma descendência "válida". Significa dizer que a rotina de limpeza na linha 4.000 do programa GENE nada tem para fazer; foi incluída unicamente para que o programa ficasse completo.

Esse tipo de programa descobrirá rapidamente uma boa solução — mas a melhor será mais lenta. Após 20.000 tentativas, é provável que a solução seja apenas um pouco melhor — se o for — do que a obtida após 10.000 tentativas.

O método de Monte Carlo é, assim, uma busca cega. Um algoritmo genético, por sua vez, mobiliza tudo o que encontra para ser orientado depois em sua busca. Padrões que contribuem para um bom desempenho são preservados e propagados por toda a população de regras. Disso resulta, na maioria dos casos, uma razoável estratégia de busca.

O programa GENE

```
10  REM  ************
11  REM  **  GENE  **
13  REM  ************
100  GOSUB 1000: REM  MONTA A MATRIZ
101 G = 0:SN = 0
105 B = SI * 1000:B$ = " "
110  INPUT "QUANTAS GERACOES ";MG
111  GOSUB 1700: REM  REGRAS INICIAIS
115  IF SN = 0 THEN  INPUT "NO INICIAL E O NUMERO ";SN
120  REM  ***LOOP PRINCIPAL***
130 G = G + 1
140  PRINT "GERACAO ";G
150  GOSUB 2000: REM  AVALIA REGRAS
160  GOSUB 2500: REM  ABANDONA AS REGRAS INADEQUADAS
170  GOSUB 3000: REM  AGRUPA AS BOAS REGRAS
180  GOSUB 3500: REM  MUTACAO
190  GOSUB 4000: REM  ORDENA
200  IF G < MG THEN 120
220  GOSUB 5000: REM  INTRODUZ NOVAS REGRAS
250  END
999 :
1000  REM  ROTINA PARA MONTAR O MAPA
1001 SI = 20:NR = 24
1010  DIM N$(20),L(20,20)
1011  DIM N(SI)
1012  DIM R$(NR),RV(NR)
1013  REM  REGRAS E VALOR DAS REGRAS
1015  FOR I = 1 TO SI
1020  FOR J = 1 TO SI
1022 L(I,J) = 1000: REM  DEFEITO
1023  IF I = J THEN L(I,J) = 0
1025  NEXT J: NEXT I
1030 NC = 0:L = 0
1032  FOR I = 1 TO NR:RV(I) = 0: NEXT I
1033  RESTORE
1040  REM  LER NOME E NUMERO DO NO
1050  READ N$,ID
1055  PRINT N$,ID
1060  IF ID <  > 0 THEN  GOSUB 1500
1070  IF ID <  > 0 THEN 1040
1080  PRINT NC;" LOCALIZACOES LIDAS"
1088  PRINT L;" MALHAS SEM DEFEITO
1090  RETURN
1100 :
1500  REM  NO INDIVIDUAL E CONECCOES
1510 NC = NC + 1
1520  IF ID <  > NC THEN  PRINT "CUIDADO NO ";ID;" NAO FUNCIONA"
1530 N$(NC) = N$
1550  REM  ***MONTA A MATRIZ DAS DISTANCIAS NODAIS***
1560  READ NI,NT
1570 L(ID,NI) = NT
1588  REM  MALHAS ZERO NAO INTERESSAM
1590 L = L + 1
1600  IF NI > 0 THEN 1550
1610  RETURN
1620 :
1700  REM  REGRAS INICIAIS SIMULADADAS
1710 S$ =  LEFT$ ("ABCDEFGHIJKLMNOPQRSTUVWXYZ",SI)
1720  FOR R = 1 TO NR
1730 R$(R) = S$
1740  PRINT S$,R
1750 I =  INT ( RND (1) * SI + 1):J =  INT ( RND (1) * SI + 1)
1755  IF I = 1 OR J = 1 GOTO 1750
1760  GOSUB 4000: REM  TROCA
1770  NEXT R
1775  RETURN
1777 :
2000  REM  ROTINA DE AVALIACAO DAS REGRAS
2010 T = 0
2020  FOR R = 1 TO NR
2030  IF RV(R) (  = 0 THEN  GOSUB 2200
2040 T = T + RV(R)
2050  NEXT R
2060 AV = T / NR: REM  VALOR MEDIO
2070  PRINT "PONTUACAO MEDIA ";AV
2080  RETURN
2100 :
2200  REM  AVALIACAO DE REGRA UNICA
2210 S$ = R$(R)
2220 P1 = SN: REM  NO INICIAL
2230 GT = 0
2240  FOR S = 1 TO  LEN (S$)
2250 P2 =  ASC ( MID$ (S$,S,1)) - 64
2260  IF P2 = SN THEN  GOTO 2290
2270 GT = GT + L(P1,P2)
2280 P1 = P2
2290  NEXT S
2300 RV(R) = GT + L(P2,SN)
2310  RETURN
2320 :
2500  REM  ROTINA PARA ABANDONAR REGRAS INADEQUADAS
2510  FOR R = 1 TO NR
2515  IF RV(R) < B THEN B = RV(R):B$ = R$(R)
2520  IF RV(R) > AV  + ( INT ( RND (1) * 100 + 1)) > B8 THEN R$(R) = " ":
    RV(R) = 0
2530  NEXT R
2540  RETURN
2550  REM  VALORES MENORES SAO PREFERIVEIS
2560 :
3000  REM  ROTINA DE DIVISAO DE GENE
3010  FOR R = 1 TO NR
3020  IF RV(R) > 0 THEN  GOTO 3120
3030 R1 =  INT ( RND (1) * NR + 1): IF RV(R1) (  = 0 THEN  GOTO 3030
3050 R2 =  INT ( RND (1) * NR + 1): IF RV(  ) (  = 0 THEN  GOTO 3050
3070  REM  PARENTES ESCOLHIDOS
3075 P2 =  INT ( RND (1) * (SI - 1) + 1)
3080 P1 =  INT ( RND (1) * SI + 1): REM  DIV.SAO DE PONTOS
3090 S$ =  MID$ (R$(R1),P1,P2)
3100  GOSUB 3300: REM  DIVIDE O RESTANTE
3110 R$(R) = S$
3120  NEXT R
3140  RETURN
3150 :
3300  REM  ROTINA DE DIVISORA DE GENE
3310  FOR S = 1 TO SI
3320 N(S) = 0: NEXT S
3330  FOR S = 1 TO  LEN (S$)
3340 SX =  ASC ( MID$ (S$,S,1)) - 64
3350 N(SX) = N(SX) + 1
3360  NEXT S
3370  FOR S = 1 TO  LEN (R$(R2))
3380 SX =  ASC ( MID$ (R$(R2),S,1)) - 64
3390  IF N(SX) ) 0 THEN  GOTO 3420
3400 S$ = S$ +  MID$ (R$(R2),S,1)
3410 N(SX) = N(SX) + 1
3420  NEXT S
3440  RETURN
3450 :
3500  REM  ROTINA DE MUTACAO
3510  FOR R = 1 TO NR
3520  IF ( RND (1) * 10) ) 8 THEN  GOTO 3580
3522 S$ = R$(R)
3525  FOR I = 1 TO 7
3530 R1 =  INT ( RND (1) * SI + 1): IF R1 = 1 THEN  GOTO 3530
3540 R2 =  INT ( RND (1) * SI + 1): IF R2 = 1 THEN  GOTO 3540
3560 I = R1:J = R2: GOSUB 4000: REM  TROCAR
3562  IF I = 1 OR J = 1 THEN 3530
3565  NEXT I
3570 R$(R) = S$
3575 RV(R) = 0
3580  NEXT R
3590  REM  TROCANDO SOMENTE AGORA
3595  REM  TAMBEM NECESSITA DE INVERSAO
3600  RETURN
3620 :
4000  REM  ROTINA ORDENADORA
4010  RETURN
4020  REM  INTRODUZ AGORA
4040 :
5000  REM  SAIDA DOS RESULTADOS
5010  PRINT "OS CAMINHOS SAO:"
5020 BB = SI * 1000
5030 B = 0
5040  FOR I = 1 TO NR
5050  IF RV(I) = 0 THEN 5100
5060  PRINT I,RV(I)
5070  PRINT R$(I)
5080  IF RV(I) < BB THEN BB = RV(I):B = I
5100  NEXT I
5105  GET A$: IF A$ = " " THEN 5105
5110  PRINT
5120  PRINT "O MELHOR E :"
5130  PRINT R$(B),B
5131 S$ = R$(B): GOSUB 6400
5133  PRINT "DISTANCIA = ";RV(B)
5134  GET A$: IF A$ = " " THEN 5134
5135  PRINT "O MELHOR DE TODOS :"
5136  PRINT B$
5140 S$ = B$: GOSUB 6400
5143  PRINT "DISTANCIA = ";B
5144  GET A$: IF A$ = " " THEN 5144
5145  PRINT
5150  RETURN
5160 :
6000  REM  ***TROCA DOIS CARACTERES EM S$***
6040  IF I > J THEN T = I:I = J:J = T
6050 X$ =  MID$ (S$,I,1)
6060 Y$ =  MID$ (S$,J,1)
6070 S$ =  LEFT$ (S$,(I - 1)) + Y$ +  MID$ (S$,I + 1)
6080 S$ =  LEFT$ (S$,(J - 1)) + X$ +  MID$ (S$,J + 1)
6090  RETURN
6100 :
6400  REM  ***VIAGEM***
6430  PRINT " 0 ";N$(SN)
6440  FOR I = 1 TO  LEN (S$)
6450 N =  ASC ( MID$ (S$,I,1)) - 64
6460  IF N <  > SN THEN  PRINT I;" ";N$(N)
6470  NEXT I
6480  PRINT I;" ";N$(SN)
6490  RETURN
6500 :
8000  REM  DADOS PARA O MAPA PLANETARIO
8010  DATA  HELIOSOL, 1
8020  DATA  2,4,17,30,20,80, 0,0
8030  DATA  MIKE,2
8040  DATA  1,4, 3,6, 0,0
8050  DATA  APHRODITE,3
8060  DATA  2,6, 5,7, 0,0
8070  DATA  LUNA, 4
8080  DATA  5,1, 6,10, 0,0
8090  DATA  TERRA FIRMA, 5
8100  DATA  3,7, 7,8, 6,8, 4,1, 0,0
8110  DATA  DEMON KINGDOM, 6
8120  DATA  5,9, 7,1, 9,12, 8,12, 5,10, 0,0
8130  DATA  PHOBIA, 7
8140  DATA  5,8, 9,13, 8,11, 6,1, 0,0
8150  DATA  EUREKA, 8
8160  DATA  6,12, 7,11, 9,1, 10,16,11,25, 0,0
8170  DATA  GALILEO, 9
8180  DATA  10,15, 8,1, 6,12, 7,13, 0,0
8190  DATA  TITANIUM CITY, 10
8200  DATA  9,15, 12,24, 11,20, 8,16, 0,0
8210  DATA  UMBRIA, 11
8220  DATA  8,25, 10,20, 12,17, 13,28, 0,0
8230  DATA  TRIDENT, 12
8240  DATA  10,24, 14,22, 13,20, 11,17, 18,2, 0,0
8250  DATA  LIMBO, 13
8260  DATA  12,20, 14,1, 16,48, 15,23, 11,20, 0,0
8270  DATA  SUNSET STRIP, 14
8280  DATA  12,22, 13,1, 18,25, 0,0
8290  DATA  HADES, 15
8300  DATA  13,233, 16,232, 0,0
8310  DATA  TROGSTAR BETA, 16
8320  DATA  13,48, 17,64, 15,32, 0,0
8330  DATA  MAXIMA CENTAURI, 17
8340  DATA  16,64, 1,30, 19,88, 0,0
8350  DATA  POSEIDON, 18
8360  DATA  17,2, 14,25, 13,25, 0,0
8370  DATA  ULTIMA THULE, 19
8380  DATA  17,88, 20,56, 0,0
8390  DATA  OMEGA SOLARIS, 20
8400  DATA  1,80, 19,95, 0,0
8410  DATA  NOWHERE, 0
8411  END
```

DIVISOR DE ÁGUAS

Estamos concluindo esta série de artigos sobre linguagem ASSEMBLY com um breve estudo da divisão binária e da programação das saídas na tela. No resumo do curso, revemos os principais tópicos abordados.

Assim como usamos o método da multiplicação binária, também o método da divisão manual serve de modelo para a divisão binária. Considere esta divisão binária:

```
      00001110 r00          quociente
1011)10011010               dividendo
   -1011                    subtrair divisor
    10000
   -1011                    subtrair divisor
     1011
   -1011                    subtrair divisor
      00                    não há resto
```

A essência desse método é a subtração repetida do divisor a partir dos bits superiores do dividendo. Dependendo do resultado dessa operação, transporta-se um 0 ou um 1 para o quociente. O resto é o resultado da última subtração de um divisor.

As várias maneiras de implementar esse algoritmo em linguagem ASSEMBLY não são tão evidentes como no caso da multiplicação. Mas também aqui a versão do Z80 usa a potência e a flexibilidade de seus registradores de 16 bits, enquanto o 6502 deve tomar e transportar 8 bits por vez. Estudaremos ambas as versões; o divisor fica no endereço representado pelo rótulo DIVSR, o dividendo em DVDND, o quociente em QUOT e o resto em RMNDR.

Observe que, em ambos os processadores, quando se subtrai o divisor do dividendo parcial, obtendo resultado negativo, deve-se restaurar o dividendo, somando-se outra vez o divisor. A versão do 6502 trabalha com o PSR (Processor Status Register, "registrador de estado do processador") após a subtração do divisor: há rotação do flag de transporte para o quociente, mas preserva-se também seu estado para indicar o resultado da subtração. Conseqüentemente, o PSR é introduzido na pilha antes da rotação e retirado logo depois, restaurando assim o transporte a seu estado pós-subtração.

Já examinamos, portanto, as quatro operações aritméticas, o que é importante como exercício de programação, pelo discernimento que traz sobre os processos da máquina; mas é desnecessário inventar combinações com a aritmética de 1 ou 2 bytes, já que há anos publicam-se tais rotinas em livros e revistas.

DIVISÃO DE 16 BITS POR 8 BITS

Z80			6502		
START	LD	A,(DIVSR)	START	LDA	#$00
	LD	D,A		STA	QUOT
	LD	E,$00		STA	RMNDR
	LD	HL,(DVDND)		LDX	#$08
	LD	B,$08		LDA	DVDHI
LOOP0	AND	A		SEC	
	SBC	HL,DE		SBC	DIVSR
	INC	HL	LOOP0	PHP	
	JP	P,POSRES		ROL	QUOT
NEGRES	ADD	HL,DE		ASL	DVDLO
	DEC	HL		ROL	A
POSRES	ADD	HL,HL		PLP	
	DJNZ	LOOP0		BCC	CONT1
	LD	(QUOT),HL		SBC	DIVSR
	RET			JMP	CONT2
DIVSR	DB	$F9	CONT1	ADC	DIVSR
DVDND	DW	$FDE8	CONT2	DEX	
QUOT	DB	$00		BNE	LOOP0
RMNDR	DB	$00		BCS	EXIT
				ADC	DIVSR
				CLC	
			EXIT	ROL	QUOT
				STA	RMNDR
				RTS	
			DIVSR	DB	$F9
			DVDLO	DB	$E8
			DVDHI	DB	$FD
			QUOT	DB	$00
			RMNDR	DB	$00

Saída na tela

Até agora usamos a RAM e a CPU para os cálculos, deixando os resultados de nossas operações em algum lugar da RAM para serem examinados manualmente por meio de um programa monitor. Esse método não é satisfatório; mas, sem conhecer a aritmética e as chamadas de sub-rotina, seria impossível obter saídas na tela com a linguagem ASSEMBLY.

A maioria dos micros tem tela de memória mapeada, ou seja, uma área da RAM dedicada a manter uma imagem na tela. A tela de vídeo compõe-se de pontos, ou pixels, que estão liga-

dos ou desligados, e que podem, portanto, ser representados por 1 ou 0 binários. Todo o conteúdo da tela é considerado como um "mapeamento", formado de pontos, dos bits que compõem a RAM de tela. Infelizmente, embora o Spectrum e o Commodore 64 usem essa técnica de mapeamento, não o fazem de forma direta. Para nossos propósitos, o método mais simples seria dividir cada linha da tela em pixels representados por bytes, numerados consecutivamente da esquerda para a direita, com o byte mais à esquerda numa linha seguindo o byte mais à direita da linha anterior. Por várias razões isso não ocorre em nenhuma dessas máquinas. Vamos considerar cada caso isoladamente.

A tela do Spectrum fica sempre no modo de alta resolução, e reserva-se uma determinada área da memória para seu mapeamento. Este, no entanto, é complexo, pois a tela se divide em três blocos horizontais de oito linhas de impressão, e cada linha divide-se, por sua vez, em oito linhas de pixels. O endereçamento dos bytes que as contêm se faz seqüencialmente dentro de cada linha, mas não entre elas. O Commodore 64 não segue esse padrão, mas tem igual complexidade. Por ora, será mais fácil entender o processo se nos limitarmos a produzir caracteres ASCII na tela.

Isso é algo que a máquina faz o tempo todo e, portanto, há, para esse fim, rotinas de código de máquina gravadas em ROM. Com uma descrição detalhada de sua operação, podemos chamar essas rotinas a partir de nossos programas em ASSEMBLY. O que precisamos saber é o endereço de chamada, os registradores de comunicações e algumas rotinas preparatórias.

No Spectrum não há rotinas preparatórias, e o registrador de comunicações é o acumulador, que deve conter o código ASCII do caractere a ser produzido. Basta apenas emitir a instrução RST $10 para que o caractere cujo código está no acumulador seja mostrado na tela, na posição atual do cursor. O código de operação RST (restart, "recomeçar") é um comando característico do Z80: trata-se de uma instrução de desvio de 1 byte, que opera na página zero, e utiliza um entre oito operandos possíveis — $00, $08, $10, $18 etc. até $38. Cada uma dessas posições indica o endereço inicial de uma rotina em ROM, situada em algum lugar da página zero.

Essas rotinas destinam-se em geral a processar entradas e saídas, e as chamamos por intermédio da instrução RST, em vez de fazê-lo diretamente pelo endereço. Isso garante maior velocidade (é mais rápido usar RST do que CALL, a nível de CPU) e também a compatibilidade do programa. Se todo programador de Z80 souber que RST $10 chama a rotina PRINT em qualquer máquina com Z80, ninguém vai precisar se preocupar em saber onde o analista de sistemas colocou a rotina PRINT, e ele ficará livre para colocá-la onde quiser, desde que a página zero seja estruturada de forma que as posições RST direcionem os programas para os endereços iniciais das rotinas.

No Commodore 64, o código ASCII no acumulador e o comando JSR $FFD2 geram o caractere na posição atual do cursor. Esta é a rotina CHKOUT, documentada no guia de referência do programador. Portanto, geralmente esse é o sistema usado de rotinas em ROM, e demonstra o princípio dos registradores de comunicações. A comunicação entre o programa principal e uma sub-rotina de entrada, por exemplo, pode passar certo caractere de um periférico para a CPU através do acumulador. Mesmo quando não há passagem de informações, um código de erro pode perfeitamente retornar da sub-rotina através de um dos registradores. Esse tipo de passagem está documentado nos muitos manuais de consulta hoje disponíveis, dedicados a equipamentos específicos. Concluímos este curso com um resumo de vários aspectos da programação em linguagem ASSEMBLY e em código de máquina.

Em resumo

Começamos o curso com uma visão geral do código de máquina, tentando desmitificá-lo e colocá-lo no contexto como apenas uma das linguagens utilizadas por programadores (e computadores). Vimos como a mesma seqüência de bytes na RAM é interpretada ora como um string de dados ASCII, ora como uma linha de um programa em BASIC, ora como uma sucessão de endereços de 2 bytes, e de novo como uma seqüência de comandos em código de máquina. Alguns minutos de experiências com um programa monitor em código de máquina bastam para convencer de que certas seqüências de bytes podem ser consideradas como três seqüências de instruções totalmente diferentes, mas válidas — dependendo de se começar a separação pelo primeiro, segundo ou terceiro byte. Nada instrínseco ao código impede que isso ocorra e, para a CPU, é indiferente executar um programa que o usuário escreveu ou alguma versão deturpada dele acidentalmente transportada para a memória.

Consideramos então a organização da memória e as convenções comuns de endereçamento. Para compreendê-las, estudamos a aritmética binária, o que de imediato delimitou nossa visão da CPU — nos processadores de 8 bits, temos de nos restringir, exceto em circunstâncias especiais, ao alcance de um só byte (ou seja, a faixa dos números decimais entre 0 e 255). Em vista da adequação da aritmética binária, evidenciaram-se as limitações do sistema decimal

no trabalho com a linguagem ASSEMBLY. Explorando a idéia da memória paginada, vimos como o tamanho das páginas lógicas deve ser função da base numérica e, num sistema binário, isso significa que o tamanho da página é uma potência de 2. Dois elevado à oitava dá 256 — o número-chave num sistema baseado em microprocessadores de 8 bits.

Como a aritmética binária logo se tornou demasiado incômoda e propensa a erros para ser usada como sistema de numeração, passamos para a aritmética hexadecimal (de base numérica 16). Vimos como é possível representar um byte de 8 bits por dois dígitos hexadecimais, de $00 a $FF, um dígito representando o estado dos quatro bits inferiores e o outro, o dos quatro superiores. Vimos com bastante detalhe a maneira como se armazenam os programas em BASIC na área reservada a programas. Ao estudarmos a simbolização como outra forma de código de máquina, fizemos uma incursão no sistema operacional. Descrevendo os marcadores de fim de linha, vimos como o interpretador BASIC reconhece onde termina um trecho de programa e outro começa. Também introduzimos a convenção de endereço baixo-alto, bem como o conceito de endereçamento indireto.

Daí passamos diretamente para a linguagem ASSEMBLY. Partimos de operações primitivas da CPU, comandadas pelos códigos de operação de 8 bits que constituem suas instruções de programação. Vimos em seguida os códigos mnemônicos do ASSEMBLY. Uma vez alcançado esse estágio, ficou claro que programar em código de máquina (ASSEMBLY) ou BASIC é tão somente programar, importando apenas resolver o problema lógico antes de nos preocuparmos com a codificação da solução. E solucionar problemas foi o tema central do curso; mas a falta de clareza de alguns conceitos do ASSEMBLY desviaram nossa atenção no sentido de eliminar a confusão que atrapalha a maioria das pessoas em seu primeiro contato com as linguagens computacionais de baixo nível.

O curso prosseguiu ensinando como carregar e executar programas em código de máquina em computadores que normalmente usam o BASIC. Examinamos as variáveis e os indicadores do sistema operacional, aprendendo como "roubar" espaço do BASIC. Passamos brevemente pela arquitetura dos microprocessadores Z80 e 6502 e começamos a escrever programas em ASSEMBLY que manipulavam a memória e o acumulador. Nesse ponto introduzimos os pseudocódigos de operação, que nos aproximaram da maneira real de programar e nos afastaram do código de máquina, da montagem manual e do trabalhoso detalhamento da programação de baixo nível.

Devido à necessidade de estruturas lógicas, passamos a considerar o PSR (Processor Status Register, "registrador de estado do processador"). Sua função de registrar os resultados das operações da CPU foi apresentada com uma introdução à aritmética binária, usando a instrução ADC ("soma com transporte"). A importância do PSR e do flag de transporte ficou esclarecida. Daí em diante, o curso concentrou-se no PSR e nas instruções a ele associadas.

Examinamos brevemente os vários modos de endereçamento: deu-se mais atenção ao indexado, devido à sua importância na manipulação de loops, listas e tabelas. Logo se evidenciou a necessidade de um tipo de instrução para alterar o fluxo de controle de um programa; e começamos a examinar as instruções de desvio condicional, ainda explorando o potencial dos modos de endereçamento direto e indireto. Com os desvios condicionais, aritmética básica e estruturas do tipo matriz, obtivemos o esqueleto de qualquer linguagem de programação, restando-nos preenchê-lo com a prática e a investigação sistemática. Examinamos o método de chamada e retorno de sub-rotinas, também como uma forma de apresentar a última área inexplorada do sistema operacional — a pilha. Vendo como ela funciona, para que serve e como a usamos, acrescentamos alguns novos recursos ao nosso repertório de programação em código de máquina. Um exame mais detalhado dos registradores da CPU e de suas interações apresentou novas possibilidades de manipulação da memória e do processador.

Por fim, com um conhecimento operacional da arquitetura do microprocessador e um vocabulário de códigos de operação, abordamos a aritmética binária, incluindo as singularidades da subtração e do complemento de 2, bem como as complexidades da multiplicação e da divisão.

Respostas dos exercícios anteriores

A) A solução de execução mais rápida é uma rotina escrita especificamente para multiplicandos de 16 bits, semelhantes à rotina para 8 bits publicada no último artigo. Por outro lado, dividindo a multiplicação de 16 bits em duas multiplicações de 8 bits (multiplicador) pelo byte inferior, depois pelo superior, podemos chamar duas vezes a rotina de 8 bits, ajustar o transporte do byte inferior e armazenar o resultado nos bytes do produto.

B) Uma rotina de multiplicação que usa a adição repetida consiste simplesmente num loop cujo contador é o valor do multiplicador; cada vez que se executa o loop, soma-se o multiplicando ao produto.

ROR — ROTAÇÃO PARA A DIREITA — 6502

Registrador — 6A (1 byte)

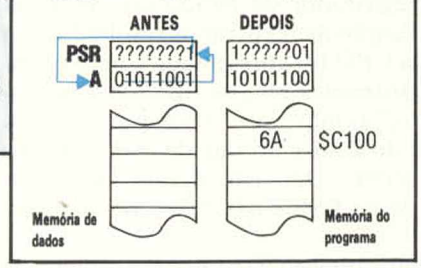
O conteúdo do byte fez rotação de 1 bit para a direita através do transporte.

EFEITO SOBRE O PSR

	S	V		B	D	I	Z	C
MSB	X						X	X	LSB

Exemplo:

LOCALIZAÇÃO	CÓDIGO DE MÁQUINA	LINGUAGEM ASSEMBLY
C100	6A	ROR A



RR — ROTAÇÃO PARA A DIREITA — Z80

Registrador CB (2 bytes)

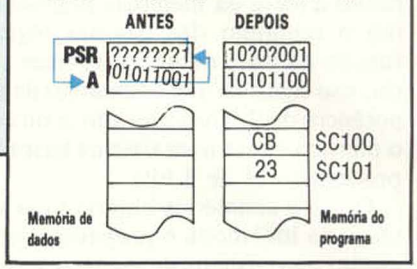
O conteúdo do byte fez rotação de 1 bit para a direita através do transporte.

EFEITO SOBRE O PSR

	S	Z		H		V	N	C
MSB	X	X		0		X	0	X	LSB

Exemplo:

LOCALIZAÇÃO	CÓDIGO DE MÁQUINA	LINGUAGEM ASSEMBLY
C100	CB 1C	RR H



SBC — SUBTRAIR COM EMPRÉSTIMO — 6502

Absoluto - ED (3 bytes)

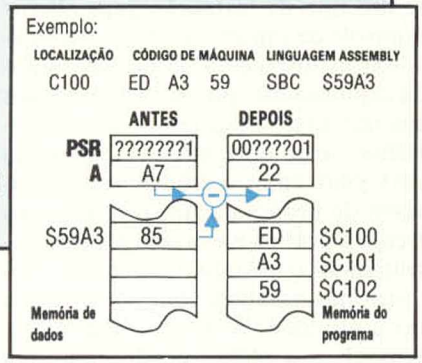
Subtrai-se do acumulador o conteúdo da posição da memória; o transporte mostra empréstimo.

EFEITO SOBRE O PSR

	S	V		B	D	I	Z	C
MSB	X	X					X	X	LSB

Exemplo:

LOCALIZAÇÃO	CÓDIGO DE MÁQUINA	LINGUAGEM ASSEMBLY
C100	ED A3 59	SBC $59A3



SBC — SUBTRAIR COM EMPRÉSTIMO DW — Z80

Imediato — DE (2 bytes)

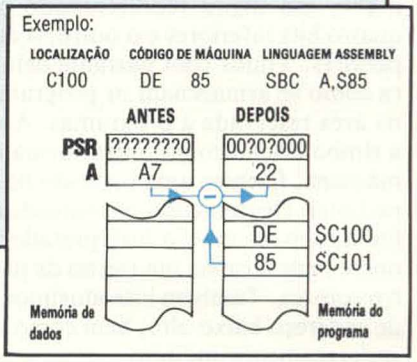
Subtrai-se do acumulador o conteúdo do byte que se segue aos códigos de operação. accumulator.

EFEITO SOBRE O PSR

	S	Z		H		V	N	C
MSB	X	X		X			X	X	LSB

Exemplo:

LOCALIZAÇÃO	CÓDIGO DE MÁQUINA	LINGUAGEM ASSEMBLY
C100	DE 85	SBC A,$85



ASL — DESLOCAMENTO ARITMÉTICO PARA A ESQUERDA — 6502

Registrador — 0A (1 byte)

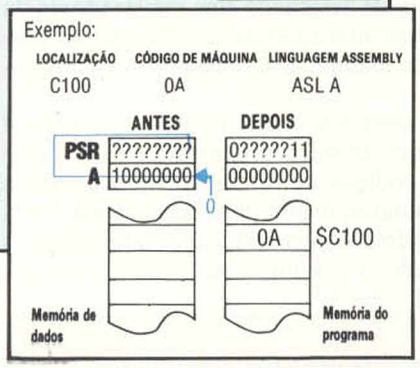
O conteúdo do byte desloca-se 1 bit para a esquerda através do transporte; entra 0 no bit LSB.

EFEITO SOBRE O PSR

	S	V		B	D	I	Z	C
MSB	X						X	X	LSB

Exemplo:

LOCALIZAÇÃO	CÓDIGO DE MÁQUINA	LINGUAGEM ASSEMBLY
C100	0A	ASL A



SLA — DESLOCAMENTO ARITMÉTICO PARA A ESQUERDA — Z80

Registrador — CB (2 bytes)

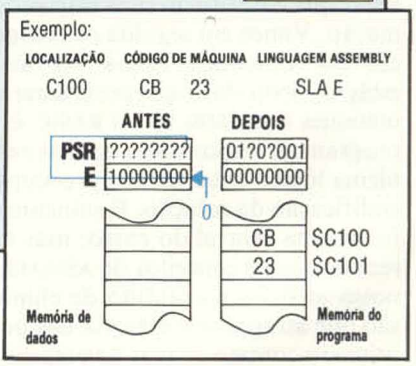
O conteúdo do byte desloca-se 1 bit para a esquerda através do transporte; entra 0 no bit LSB.

EFEITO SOBRE O PSR

	S	Z		H		V	N	C
MSB	X	X		0		X	0	X	LSB

Exemplo:

LOCALIZAÇÃO	CÓDIGO DE MÁQUINA	LINGUAGEM ASSEMBLY
C100	CB 23	SLA E



CMP — COMPARAR O ACUMULADOR — 6502

Absoluto - CD (3 bytes)

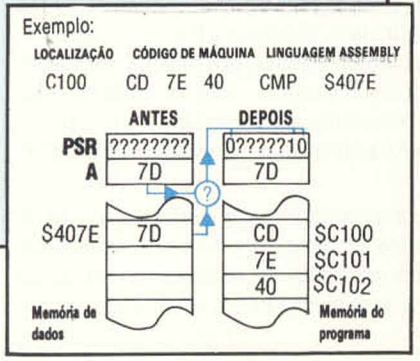
Compara-se o acumulador com o conteúdo da posição da memória.

EFEITO SOBRE O PSR

	S	V		B	D	I	Z	C
MSB	X						X	X	LSB

Exemplo:

LOCALIZAÇÃO	CÓDIGO DE MÁQUINA	LINGUAGEM ASSEMBLY
C100	CD 7E 40	CMP $407E



CP — COMPARAR O ACUMULADOR — Z80

Imediato — FE (2 bytes)

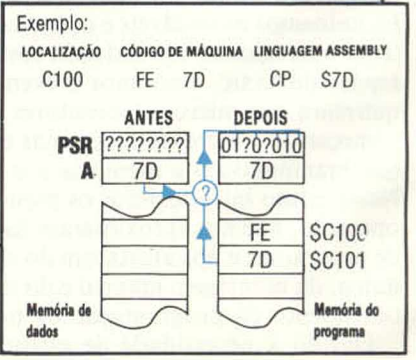
Compara-se o acumulador com o conteúdo do byte que se segue ao código de operação. accumulator.

EFEITO SOBRE O PSR

	S	Z		H		V	N	C
MSB	X	X		X		X	1	X	LSB

Exemplo:

LOCALIZAÇÃO	CÓDIGO DE MÁQUINA	LINGUAGEM ASSEMBLY
C100	FE 7D	CP $7D



MICROCOMPUTADOR
CURSO PRÁTICO

MICROCOMPUTADOR
CURSO PRÁTICO

EDITORA rioGráfica
LAZER, CULTURA E INFORMAÇÃO

rioGráfica

CONSELHO DE ADMINISTRAÇÃO
Roberto Irineu Marinho
João Roberto Marinho
José Roberto Marinho
Oscar Neves
Danilo Esteves Costa

Diretoria
Oscar Neves
Danilo Esteves Costa
João J. Noro
Marcos March
Nilo Sérgio de Almeida

GRUPO EDUCAÇÃO E CULTURA

Divisão Comercial

Diretor
Jaime Rodrigues
Gerente de Planejamento
João de Azeredo Coutinho
Gerente de Vendas
Rubens Barbosa

Editorial

Diretor
João J. Noro
Editor-chefe
Maurício Rittner
Diretor de Arte
Adalberto Cornavaca
Administração Editorial
Heitor de Souza Paixão

Editor
Jesse Navarro
Editor-assistente
Cláudio Marcondes
Coordenadora de Revisão
Isis Augusta Loyolla
Revisores
Cacilda Guerra
Carlos Eduardo Russo Tranjar
Luiz Vicente Vieira Filho
Maria da Graça Mendonça Couto
Sérgio Roberto Torres
Chefe de Arte
Bonifácio Duardes Miranda
Chefe de Estúdio
José Yuji Kuribayashi
Assistentes de Arte
Miguel Luis Escámez Simón
Edemilson R. da Silva
Wagner Celestino

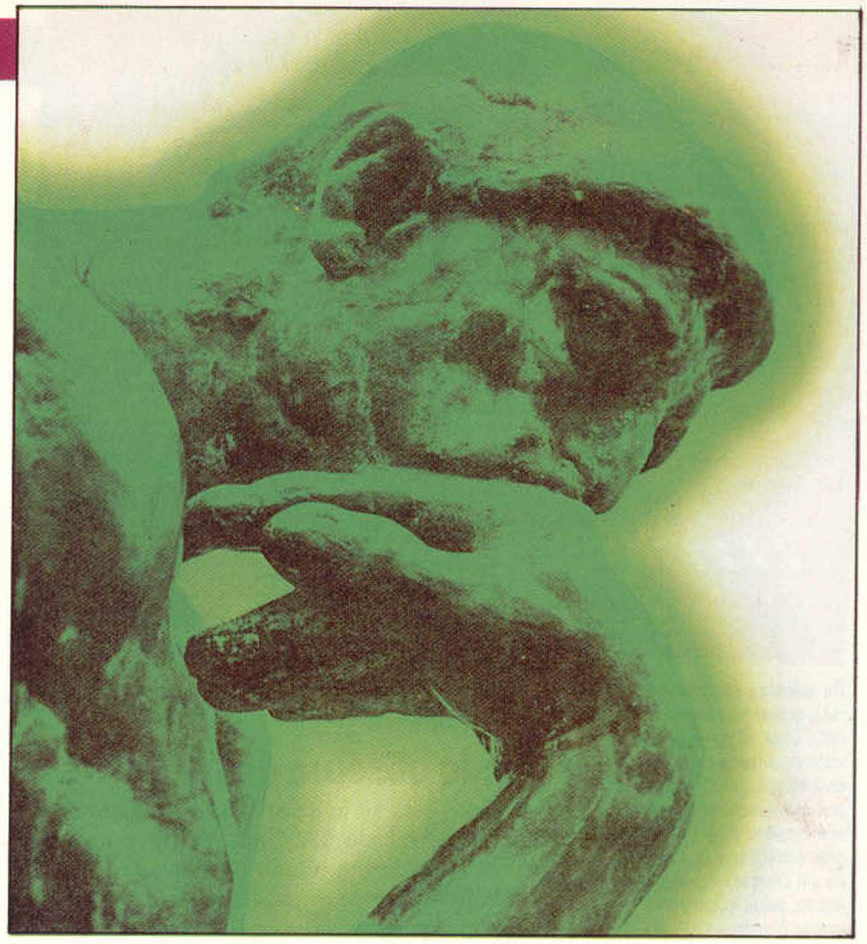
Colaboradores
Editores-assistentes:
Isa Mara Lando
Luiz Carlos Pizarro Marin
Texto:
Carlos Eduardo Silveira Matos
José Antônio Arantes
Paulo Brito
Solange A. Menezes
Preparação de Texto:
Luiz Carlos Cardoso
Tradução:
Fernando Dorfman Knijnik
Luiz Roberto de Godoy Vidal
Maria Clara Cescato
Maria Ester Menezes Martinho
Newton Roberval Eichemberg
Pesquisa:
Stella Maria Quentel
Fotos Nacionais:
Fernando Scavone
Jorge's Estúdio
Salvador de Rosa
Assessoria:
Geraldo Coen
Gustavo José Ferreira Grubba
José Rubens Salles Toledo
Lauro de Lauro
Lucas Kenichi Shirahata
Pierluigi Piazzi

© APSIF Copenhagen, 1984
© Orbis Publishing Ltd., 1984
© Editora Rio Gráfica Ltda., 1985, para a língua portuguesa

Composição: ERG Ltda. e AM Produções Gráficas Ltda.
Impressão: JBIG

VISÃO DO FUTURO

Não conhecemos ainda o efeito das máquinas dotadas de inteligência criativa sobre o homem. Mas é certo que terá um impacto decisivo sobre as gerações futuras — de seres humanos e de computadores.



O que nos reserva o futuro
Os progressos da biotecnologia poderão nos abrir a perturbadora perspectiva dos biônicos, híbridos homem-máquina, combinando os processos intuitivos do cérebro humano com o potencial lógico e matemático dos microprocessadores. Se for esse o caso, assistiremos à evolução humana sair das mãos da natureza e ficar sob o controle de homens-máquina capazes de auto-reprodução seletiva.

Comparando o desenvolvimento em computação a uma escada, podemos, num sentido amplo, distinguir quatro degraus, de crescente complexidade. No primeiro, temos as tarefas corriqueiras do guarda-livros, que os computadores desempenham muito bem. Um degrau acima, estão os programas que ajudam as pessoas a tomarem decisões inteligentes — sistemas de revisão financeira e planilhas, por exemplo. Essas ferramentas precisam ser flexíveis, uma vez que não se pode conhecer de antemão as exigências do usuário.

Ao nível do terceiro degrau, encontramos a aplicação de habilidades derivadas daquelas dos seres humanos. É aí que se concentram as aplicações da IA (Inteligência Artificial), agora e no futuro imediato. Um software como esse tem exemplo no sistema de habilidade Prospector, que codificou os procedimentos de vários geólogos. A aplicação prática dessas normas de trabalho à prospecção culminou com a descoberta de grandes depósitos de molibdênio no Estado de Washington, até então desconhecidos. Depois disso, outro depósito foi descoberto no Canadá.

As façanhas do Prospector passaram para a mitologia da IA, e de fato impressionam. Mas não nos levam ao quarto degrau da escada: o das máquinas dotadas de inteligência criativa. Para compreender isso, precisamos fazer uma contraposição entre produtividade e criatividade.

A produtividade depende da obediência a regras — e nisso os computadores são excelentes. Eles podem se mostrar bem mais produtivos que as pessoas, mas daí a torná-los criativos vai grande distância, constituindo tarefa extremamente difícil. Para atingirem o atributo de criativos, os programas de computador precisarão ser capazes de elaborar suas próprias regras. E, de fato, alguns pesquisadores que trabalham nas fronteiras da IA estão tentando conseguir nada menos que isso.

Um programa que deu um passo na direção do quarto degrau é o Eurisko, projetado por Doug Lenat, da Universidade de Stanford. Trata-se de um programa de descoberta que tem sido aplicado em vários domínios, desde jogos de batalha naval até desenhos de circuitos VLSI (Very Large Scale Integration, "integração em escala muito ampliada"). O Eurisko parte de uma coleção de regras e conceitos heurísticos e os aplica aos domínios escolhidos. Muita coisa no programa é padronizada, consistindo a novidade no fato de ele poder modificar sua própria heurística. Isso dá ao sistema um grande poder, visto ser ele capaz de adaptar e especializar suas regras genéricas, a fim de lidar com situações novas.

O Eurisko fez uma descoberta que mais tarde resultou numa patente, e isso dificilmente pode ser considerado trivial. Trata-se da descoberta de um novo desenho para o circuito lógico em 3 D (uma porta NÃO-E/OU), sobre o qual jamais alguém no Vale do Silício — ou em qualquer outro lugar — havia pensado. No entanto, originou-se da aplicação de uma única regra.

Quando o Eurisko foi aplicado ao projeto do VLSI, ele já possuía uma regra heurística, proveniente de tarefas anteriores, que afirmava o seguinte: "Se um conceito é interessante, tente torná-lo ainda mais". Aplicada a um dispositivo em 2 D, tal regra leva a uma versão mais simétrica, em 3 D, desse dispositivo — recentemente fabricado com sucesso. Um componente em 3 D, depois de isolados os problemas de manufatura, pode ser compactado mais densamente, oferecendo maior capacidade.

A criatividade é vista como o ponto máximo da inteligência humana, e encontra-se mergulhada em nebulosas explicações envolvendo intuição e lampejos; mas o Eurisko pode nos dar a necessária pausa para pensar. Constitui exemplo de autêntica descoberta resultante da aplicação

Da máquina ao super-homem HAL, o computador do filme *2001: Uma odisséia no espaço*, visto aqui numa cena de sua continuação, *2010, o ano em que faremos contato*, começou sua existência com capacidade para sintetizar a fala e dotado de um complexo conjunto de regras, pelas quais podia avaliar o sucesso de uma missão. Tais características são hoje implementadas em proporção cada vez maior no hardware e no software. No livro *2010*, Arthur C. Clarke levou o conceito de inteligência da máquina vários passos adiante, fazendo HAL transcender seu status mecânico e se tornar uma figura super-humana, questionando o pressuposto de que as máquinas não podem evoluir independentemente de seus criadores.

de uma só regra. O computador criativo encontra-se mais perto de nós do que pensa a maioria das pessoas.

O lado obscuro da IA

Até aqui, esta série de artigos concentrou-se na face aceitável da IA, mas existe aquilo que alguns consideram o lado obscuro da IA. Independentemente de quanto sejam impressionantes os progressos na pesquisa da IA, podemos não gostar de algumas de suas aplicações — especialmente se levarmos em conta que boa parte do trabalho de pesquisa da IA depende da ajuda financeira militar. Mais de metade de suas aplicações a curto prazo visa o campo de batalha. Essas aplicações incluem:

- Submersíveis inteligentes;
- Munições "perspicazes";
- Mísseis de cruzeiros;
- Tanques autodirigidos;
- Sistemas de sonar baseados em conhecimentos;
- Torpedos que procuram o alvo;
- Sistemas de produção de imagens de radar

e muitos outros dispositivos dos quais poucos de nós já ouviram falar.

Mais alarmante, porém, é o que algumas pessoas imaginam como resultado de pesquisas excessivamente bem-sucedidas com vistas à criação de MUIs (Máquinas Ultra-Inteligentes). Essas pessoas vislumbram um futuro dominado pelas MUIs. Incapazes de realmente "pensar", as máquinas poderão, no entanto, fazer — e bem melhor que o ser humano — quase tudo o que exige atitude racional. Na matemática e nas ciências naturais, irão muito além de nós. Na indústria, a persistir o atual panorama, elas excederão a tal ponto os administradores humanos que a execução de toda a economia estará sob seu controle. Em resumo: seremos intelectualmente inferiores a elas.

A idéia de seres "biônicos" — híbridos, meio homens meio máquinas — tem sido uma das favoritas entre os escritores de ficção científica. Mas recentemente ela se transferiu do campo da imaginação para a área da conjectura científica a longo prazo. A ascendência de duas áreas da pesquisa científica aparentemente incompatíveis — a engenharia genética e a engenharia do conhecimento — é em grande parte responsável por isso.

A engenharia genética está interessada na manipulação do código genético na matéria viva, de modo a "programar" melhorias genéticas em gerações sucessivas. Os engenheiros geneticistas já aperfeiçoaram técnicas que lhes permitem enxertar características desejáveis em microorganismos. Com isso, conseguem-se drogas que antes atingiam custos proibitivos ou eram simplesmente impossíveis de fabricar em larga escala. Espera-se que, num futuro próximo, a engenharia genética desenvolva os chamados biochips. Programando a ação das enzimas que dividem e reconstituem moléculas, será possível fazer os circuitos lógicos *crescerem* a partir de moléculas de proteínas, o que resultará numa formidável redução de tamanho. A idéia de utilizar material vivo para transportar mensagens elétricas codificadas não é nova; o sistema nervoso funciona assim.

Atualmente já se consegue alterar a composição genética da carne para torná-la mais resistente ao ataque de fungos. O próximo passo será utilizar essa técnica para erradicar doenças genéticas dos seres humanos, como a síndrome de Down, por exemplo. Assim que forem dados os primeiros passos na manipulação dos genes humanos, eventualmente surgirão outros experimentos — digamos, os cientistas poderão tentar enxertar no cérebro humano um bio-ROM contendo um banco de dados. Os cientistas já obtiveram algum sucesso em enxertar dispositivos elétricos no cérebro. Uma estudante surda de Oxford recebeu em seu cérebro, nas áreas responsáveis pela audição, a saída elétrica de um microfone. Após breve período de treinamento, ela conseguiu extrair sentido dos sinais recebidos — o suficiente para "ouvir" sons. Em poucas palavras: parece que o cérebro é capaz de aprender a se colocar como interface em relação a fontes elétricas externas.

Algumas pessoas propõem que, num futuro distante, nossos descendentes evolutivos possam ser algum tipo de híbrido homem-máquina altamente avançado. Como o processo evolutivo parece fazer uso de qualquer material à mão — os pulmões, por exemplo, mudam em órgão de flutuação —, parece razoável supor que esses organismos híbridos venham a abrigar, em seus corações, um cérebro humano circundado por várias camadas de tecnologia avançada. De fato, em sua forma atual, o cérebro humano apresenta várias camadas, que se desenvolveram durante a nossa evolução.

As aplicações dessas duas novas tecnologias são ainda muito incertas. E a realização ou não das hipóteses aqui esboçadas parece depender mais do enfoque social (constituem algo desejável?) do que da praticabilidade técnica.

SUMÁRIO

VOLUME 2

APLICAÇÕES

Robôs domésticos. 317
Produtores de melodia. 337
Nova música eletrônica. 357
O protocolo MIDI. 382
MIDI na prática. 397
Música em software. 417
Robôs na trilha. 422
Gran finale. 437
Sensores robóticos. 442
Visão robótica. 462
Comando mental. 477
O passo do robô. 490
Perfil do futuro. 497
Sob medida. 508
Busca inteligente. 517
Enciclopédia eletrônica. 523
Conversa de robô. 537
Grande Mestre. 540
Consulte o especialista. 563
Operário padrão. 568
Robôs para montar. 577
Reconhecimento visual. 589
Aprendendo a aprender. 597
Visão do futuro. 605

CHIPS & BYTES

Na velocidade da luz. 330
História do microprocessador. 353
Entradas e saídas. 362
Tal e qual. 377
Ponta-de-lança. 404
Redes locais. 430
Escher eletrônico. 435
Primeiras letras. 440
Solução para compatibilidade. 482
Força bruta. 488
A luz fantástica. 502
Nova aprendizagem. 520
Biorritmo. 534
Teledados. 548
Guerra de imagens. 557
Saída para a expansão. 560

CIÊNCIA DA COMPUTAÇÃO

LINGUAGENS

Estrutura versátil. 320
Dupla dinâmica. 342
O GOTO AND. 372
Introdução ao ASSEMBLY. 392
Memória paginada. 412
Byte a byte. 433
Espaço reservado. 452
Transporte de valores. 472
Mudança de endereço. 493
Saltos e desvios. 512
Rotinas genéricas. 531
Rascunho eletrônico. 553
O segredo das operações. 572
Múltipla escolha. 583
Babel informática. 592
Divisor de águas. 601

PERFIL

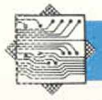
Unitron. 336
Polimax. 356
SORD. 375
Psion. 396
Cobra. 415
Microtec. 436
Proceda. 456
Sisco. 476
Labo. 496
Sid. 516
Edisa. 535
Microsoft. 556
Elebra. 576
Medidata. 596

PERIFÉRICOS

Toque de mestre. 402
Made in Brazil. 457

PROJETOS DE PROGRAMAS

Graus de precisão. 340
Controle auditivo. 379
Aventuras e simulações. 428
Centopéia famintą. 480
Base lunar. 510



RAIO X

N 684. 325
Epson PX-8. 345
Máquina de desenho. 365
Osborne Encore. 385
Canon T70. 405
Wren Executive. 425
Yamaha CX5M. 445
Sanyo MBC-550. 465
Padrão MSX. 485
Spectravídeo 318. 505
Expert. 526
RAT. 546
Atari XL. 566
Tandy Modelo 100. 586

SOFTWARE

O nome dos nomes. 328
TK! Solver. 348
Operador discreto. 350
A chave dos dados. 360
Textos em tela. 368
Criação de registros. 380
Relações produtivas. 388
Tentativa e erro. 390
Referências cruzadas. 400
Efeitos visuais. 408
Idéias em ordem. 420
Gerência eletrônica. 448
Fim das pilhas. 468
Modelos em perspectiva. 500
Pacotes financeiros. 528
Trabalho de bastidor. 544
Geradores de programas. 580

ÍNDICE

A

Acoplador acústico 198(1)
ACIA (Asynchronous Communications Interface Adaptor) 362-364, 382(2)
Acumulador 434(2)
ADA 7(1)
Adaptador para interface de comunicações assíncronas
ver ACIA
Adaptador de interface periférica
ver PIA
Alfa 2064 196(1)
Alfa 3003 **146-147**(1)
Álgebra booleana **92-94, 208-210**(1)
ALGOL (Algorithimic Language) 7(1), 593(2)
Algoritmo 8, 75(1), 320-324(2)
Alto-falante 245(1)
ALU (Arithmetic and Logic Unit) 392, **470-471**(2)
Análise
semântica 537(2)
sintática 205(1), 537(2)
Antunes, Geraldo 336(2)
APL (A Programming Language) 7(1), 594(2)
Aplicativos **291**(1)
Apple IIc **46-47**(1), 560-562(2)
Aprendizado
sistemas de 597-600(2)
Apricot **106-107**(1)
Arco 340(2)
Arquivos
indexação de registros **122-123**(1)
manipulação de 140-143(1)
seqüenciais **62-63, 90-91, 102-103**(1)
ver tb. Banco de dados
Asimov, Isaac 238(1)
ASSEMBLY 412-414, 433-434, 452-455, 493-495, 512-515, 553-555(2)
adição **472-475**(2)
controle de entrada e saída 362-364(2)
divisão binária 601-604(2)
rotinas **532-534**(2)
subtração e multiplicação **583-585**(2)
Atari XL **566-567**(2)

B

BAM (Block Availability Map) 21(1)
Bancada
teste de **192-193**(1)
Banco de dados 42-43, 225-227(1), 322, **328-329,** 517-519(2)
armazenamento 14, **20-21,** 62-63, 90-91, 102-103, 140-143(1)
gerenciamento de 117-119(1), 328-329, **380-381**(2)
organização **303-304**(1)
referências cruzadas **400-401**(2)
Banktec 76(1)
BASIC **124-125**(1)
comandos 130-131, 275(1)
compiladores 294
funções trigonométricas 340-341(2)
Bastão fotônico 137(1)
Baudot, Émile 199(1)
Bauds (medida) 188-189(1)
BBC Modelo B **285-287**(1)
BDOS (Basic Disk Operating System) 229(1), 352(2)
BEAGLE (Biological Evolutionary Algorithm Generating Logical Expressions) 597-598(2)
Bellonzi, Leonardo 296(1)
Big trak 522(2)
Bingo **253**(1)
Biorritmo **528**(2)
BIOS (Basic Input/Output System) 229(1), 352(2)
Bits 199(1), 412(2)
de paridade **292-293**(1)
de transporte 472-475(2)
Bitstik 501(2)
Blumenfeld, Joseph 296(1)
Boole, George 28, 208(1)
Braille 138(1)
Brailler, Perkins 138(1)
BrainStorm **420-421**(2)
Brother EP-44 240-241(1)
Buffer 19(1)
Busca
técnica de 529-531(2)
Bushnell, Nolan 115-116(1)
Byte 412(2)

C

C 7(1), 595(2)
CAD (Computer Aided Design) 375, 500-501(2)
Calculadora eletrônica 196(1)
Câmara **246-247**(1)
eletrônica 405-407(2)
infravermelha 443(2)
CAMS (Computer Aided Music System) 439(2)
Caneta óptica 15(1)
Canon T70 **405-407**(2)
Capacitores 245(1)
Capek, Karel 238(1)
Caracteres 211, 213(1)
Card Box 328-329, 401(1)
Carlos, Walter 339(2)
Cartucho 20(1)
CBM (Commodore Business Machine) 268-269(1)
Chave 245(1)
Cheung, Paul 521-522(2)
Chips **353-355**(2)
de interface de vídeo 408-409(2)
Chowning, John 338(2)
Cibernética 257, 280(1)
Circuito
conversor de sinais **334-335**(2)
de sincronização **410-411**(2)
elétrico 244(1)
eletrônico **168-169**(1)
integrado 36(1)
lógico **28-29,** 92-94(1)
adição **48-49**(1)
seqüencial **370-371**(2)
CLASCAL 181(1)
CMOS (Complementary Metal Oxide Semi-conductor) 20(1), 346, 407(2)
COBOL (Common Business Oriented Language) 7(1), 592-593(2)
Cobra 480 416(2)
Cobra 530 416(2)
Codificador 232-233(1)
de eixo 423(2)
Códigos **232-233**(1)
de máquina 7(1)
Comando mental **477-479**(2)
Combinações lógicas 470-471(2)
Commodore PET 175-176(1)
Commodore Plus/4 **126-127**(1)
Commodore 64 176(1), 397, 408-409(2)
Compaq Plus **266-267**(1)
Compatíveis 9-13(1)
Componentes **244-245**(1)
Computação óptica **502-504**(2)
aplicação **5-6**(1)
comércio **448-449**(2)
doméstica **277-279**(1)
educação 22-25, 30-33, 297-299(1), 440-441, 517-522, 568-571(2)
lazer 6, 24-25, 44-45, 217-219, 253, 280-281(1), 428-429, 540-543(2)
medicina 258-259(1)
música 6(1), 337-339, 357-359, 382-384, 397-399, 417-419, 437-439, 445-447(2)
processador de texto 17-19, 39, 137, 226, 241(1), 368-369, 420-421(2)
controle mental **477-479**(2)
equipamentos usados **310-311**(1)
marketing **557-559**(2)
relação homem-máquina **194-195**(1)
usuário 194-195(1)
Contato
teste de 192(1)
Controladores programáveis 16(1)
CP/M (Control Program/Monitor) 56, 135-136(1), 544-545(2)
CP/NET 545(2)
CPU (Central Processing Unit) 392-393, 450-451, **452-455,** 572-575(2)
CTS (Clear To Send) 363(2)
Curva
de Sierpinski 80-81(1)
dos flocos de neve 80-81(1)

D

D-8100 196(1)
Dados
armazenamento 14, **20-21,** 90-91, 102-103, 140-143(1)
endereços **328-329, 493-495**(2)
equipamentos 225-227(1)
Banco de 117-119, **303-304**(1), 328-329, 380-381, 400-401(2)
compiladores **342-344**(2)
teste dos 308(1)
transmissão de 198(1)

DCD (Data Carrier Detect) 363(2)
Decodificador 232-233(1)
de prioridades 292-293(1)
Default 62(1)
Deficientes físicos 137-139(1)
Demodulação 197-198(1)
Desenho 184-185, 305-307(1), **365-367,** 402-403, **500-501**(2)
Dessoldagem 128-129(1)
Detecção de transporte de dados
ver DCD
Detector
de mentiras 479(2)
de proximidade 443(2)
Diagnóstico médico 258-259(1)
Diagramas de Venn 92(1)
Diez, Ricardo 336(2)
Dilts, Roger 478(2)
Diodos emissores de luz
ver LED
DIP (Dual In-line Packages) 89(1)
Disco
diretório 43(1)
flexível 21(1)
laser 151-153(1)
óptico 502(2)
sistema operacional de 21, 42-43(1), 351-353, 544-545(2)
unidades de 460(2)
Dispositivos gráficos 15(1)
Disquete
ver Disco flexível
Documentos
transmissão via telefone **377-378**(2)
DOS (Disk Operating System) 42-43(1), 351-352(2)
Dump 185(1)

E

Educação 22-25, 30-33, 297-299(1), 440-441, 568-571(2)
EEPROM (Electrically Erasable Programmable Read Only Memory) 20(1)
Embramic 2000 139(1)
Endereçamento 493-495(2)
etiquetas de 120-121(1)
Endereços
na memória **328-329, 493-495**(2)
Eno, Brian 338(2)
EPROM (Erasable Programmable Read Only Memory) 330-331(2)
Epson PX-8 **345-347**(2)
Equações
processamento de **348-349,** 390-391(2)
Ergonomia **508-509**(2)
Erro **173-174**(1)
verificação 199(1)
Escher, M.C. 435(2)
Espionagem **77-78**(1)
Eurisko 605(2)
Expert **525-527**(2)

F

Fac-símile **377-378**(2)
Faggin, Frederico 56(1)
Fala 538-539(2)
Fatura 121(1)
Feder, Joseph 196(1)
Feigenbaum, Edward 258(1)
Ferramentas **68-69,** 192-193(1)
Figuras tridimensionais 270-271(1)
Fita cassete 20-21(1)
Flags 512-515(2)
Flip-flop
ver Circuito seqüencial
Fliperama 116(1)
Fluxograma 95, **113-114**(1)
FORTH 7(1)
FORTRAN (Formula Translator) 7, 288-290(1), 592(2)
Fotossensores 423(2)
Fregni, Edson 236(1)
Fuzil óptico 15(1)

G

Gadd, Steve 359(2)
Gaertner, Vilmar 336(2)
Galvin, Paul 256(1)
Garra jacaré 192-193(1)
Gates, Bill 556(2)
GENE (General Evolutionary Network Explorer) 598-600(2)
Glissando 383-384(2)
GPS (General Problem Solver) 258(1)
Gráficos **64-65, 79-81,** 160-161, 177-179, 282-284(1), 402-403, 408-409, 435, **500-501, 510-511**(2)
tridimensionais **190-191**(1)
Grafix 100Mx **206-207**(1)
Grafpad **305-307**(1)
Grainer, Ron 338(2)
Gravação
digital **437-439**(2)
seqüenciada 62(1)
Gravador
cassete 14(1), 457(2)
de EPROMs **330-331**(2)
Guitarra 418-419(2)

H

Hashing **122-123**(1)
HERB (Heuristic Evolutionary Rule Breeder) 598(2)
Hiller, Lejaren 338(2)
Haung, Lawrence 356(2)
Huizinga, J. 299(1)

I

I-7000PCxt **166-167**(1)
IBM PC 12, **26-27**(1), 560-562(2)
Ícones 185(1)
Impressão
bidirecional 200(1), 459(2)
sistema térmico de 240(1)
Impressora 59-61(1), 458-459, 461(2)
eletrostática 59-61(1)
matricial 16, 59, **88-89,** 120-121, 160-161, 200, **206-207**(1), 459(2)
térmica 16, 59-61(1)
tipo margarida 16, 59, **200-201**(1), 459(2)
traçadora 16(1)
Índice 122(1), 493-494(2)
Informática
política brasileira de 415-416(2)
Input/Output
ver I/O
Inteligência artificial **257-259**(1), 589-591, 605-606(2)
Interfaces 10(1), 461(2)
do usuário 509(2)
I/O (Input/Output) 362-364(2)
PIA (Peripheral Interface Adaptor) 362-364(2)
Interruptor 502-503(2)
I/O 362-364(2)
IS (Intelligent Software) **249**(1)
ISO (International Standards Organization)
ver Organização Internacional de Padronização

J

Jacquet-Droz, Pierre 237(1)
Jogos 24-25, 44-45, 82-83, 100-101, 104-105, 115-116, 124-125, 142-143, 230-231, 297, 300(1), 480-481, 510-511, 541(2)
de cartas 85(1)
de estratégia 540-543(2)
de labirinto **529-531**(2)
de simulação **164-165,** 297-299(1), **428-429**(2)
educacionais 297-299(1)
fliperama 116(1)
space invaders 116(1)
xadrez **217-219**(1), **540-543**(2)
Joypad 546-547(2)
Joysticks 16(1), 477, 546(2)

K

Karchner, John 36(1)
Karnaugh, mapas de 154-155, **168-169,** 208(1)
Kilby, Jack 36(1)
Kildall, Gary 56, 135-136(1), 353, 544(2)
Kinnear, Robin 396(2)

L

Lazer 6, 24-25, 44-45, 217-219, 253, 280-281(1), 428-429, 540-543(2)
LCD (Liquid Crystal Displays) 99, 240-241(1)
LEAF (Logical Evaluator Aid Forecaster) 598(2)
LED (Light Emitting Diodes) 245(1)
Lee, Marcos 356(2)
Levine, Steve 339(2)
Levy, David 249(1)
Linguagem 7(1), **592-595**(2)
alto nível 181(1)
baixo nível 532(2)
de máquina
ver ASSEMBLY
humana 537-539(2)
análise da 204-205(1)
LISP (List Processor) 7, 31(1), 593-594(2)
Listas 148-150(1)
manipulação de 420-421(2)
Livro-caixa 449(2)
Loftus, Elizabeth 297(1)
Loftus, Geoffrey 297(1)
LOGO 7, **30-33,** 50-53, 79-83, 132-134, 138(1), 521(2)
cálculos numéricos **110-112**(1)
comandos 32-33, 53, 71(1)
procedimentos **170-172**(1)
rotinas 50-53, 72(1)
Loop 95(1), 512-515(2)

M

McCulloch, Warren 257(2)
Macintosh **86-87**(1)
Macrocomando **202-203**(1)
Madge, Robert 249(1)
Mala direta 19(1), **468-469**(2)
Manasterski, Joseph 236(1)
Mann, Richard 36(1)
Manton, Tony 156(1)

Mapa de disponibilidade de blocos
ver BAM
Mapas K
ver Karnaugh, mapas de
Máquina
alvo 248-249(1)
código de 452-455(2)
de escrever 196, **240-241**(1), 459(2)
Martin, Allan 299(1)
Martin, George 339(2)
Maxxi 356(2)
McDermott, Eugen 36(1)
MCL (Music Composition Language) 383, 417-418(2)
Medicina 258-259(1)
Medidor analógico 192(1)
Memória 450-455(2)
acesso 18(1)
manipulação da **572-575**(2)
paginada **412-414**(2)
RAM 20(1)
ROM 20(1)
Menu **254-255**(1), 362-369(2)
pull-down 185(1)
Meteorologia 489(2)
Micon (MIDI Controller) 397-398(2)
Microfone 245(1)
Micro
acesso via telefone **197-199**(1)
compatibilidade **482-484**(2)
componentes 245(1)
expansão **560-562**(2)
modelos **9-13**(1)
Alfa 2064 196(1)
Alfa 3003 **146-147**(1)
Apple IIc **46-47**(1), 560-562(2)
Apricot **106-107**(1)
Atari XL **566-567**(2)
BBC Modelo B **285-287**(1)
Cobra 480 416(1)
Cobra 530 416(1)
Commodore PET 175-176(1)
Commodore Plus/4 **126-127**(1)
Commodore 64 176(1), 397, 408-409(2)
Compaq Plus **266-267**(1)
D-8100 196(1)
Embramic 2000 139(1)
Epson PX-8 **345-347**(2)
Expert **525-527**(2)
I-7000PCxt 13, **166-167**(1)
IBM PC 12, **26-27**(1), 560-562(2)
Macintosh **86-87**(1)
Maxxi 356(2)
MSX 482-484, **485-487**(2)
My Talking Computer **440-441**(2)
Nexus 1600 236(1)
Nexus 1684 325-327(2)
Osborne-1 385(2)
Osborne Encore **385-387**(2)
Osborne Executive 385(2)
PC-2001 436(2)
Poly 101 HS 356(2)
Sanyo MBC-550 **465-467**(2)
SB-80 376(2)
Sharp PC-1251 **66-67**(1)
Sharp PC-1500A **66-67**(1)
Spectravídeo 318 **505-507**(2)
Tandy Model 100 586-588(2)
TI99/4A 36(1)
TK 90X **186-187**, 276(1)
TRS-80 11(1)
Unitron 336(2)
Wren Executive **425-427**(2)
Yamaha CX5M **445-447**(2)
Z80 56(1)
Z8000 56(1)
portáteis **97-99**(1), 396(2)
redes de 96, **37-39**(1), **430-432**(2)
Microprocessador 56, 256(1), **353-355**, 404(2)
Microware 136(1)
MIDI (Musical Instrument Digital Interface) 337, 382-384, 397-399, 417-419(2)
Minicomputador 250(1), 476, 576, 596(2)
Minsky, Marvin 31(1)
Mixagem 419(2)
MLOGO 30(1)
Modem 10, 15, 197-198(1), 459-460(2)
MODULA 2 7, 181(1)
Modulação 197(1)
Monitor 301-302(1), 351(2)
de vídeo 16, **40-41**(1), 458(2)
Moore, George 237(1)
Moroder, Giorgio 358(2)
Motor 279(1)
Mouse **184-185**, 228(1)
MP/M (Multiprogram/Monitor) 545(2)
MSX 482-484, **485-487**(2)
Multissintetizador 418(2)
Multímetro 192(1)
Música 6, 156(1), **337-339**, **357-359**, **382-384**, 397-399, **417-419**, 437-439, 445-447(2)
My Talking Computer **440-441**(2)

N

Nexus 1600 236(1)
Nexus 1684 **325-327**(2)
Newell, Alan 258(1)
Números
aleatórios 111-112(1)
binários 413(2)
hexadecimais 414(2)

O

Ohm, George 192, 244(1)
Ohm, lei de 244(1)
Ohmímetro 192(1)
Oliveira, Sérgio Sá Moreira de 336(2)
Olivetti, Adriano 216(1)
Olivetti, Camillo 216(1)
Ondas senoidais 337, 357(2)
Organização Internacional de Padronização (ISO) 211(1)
Osborne-1 385(2)
Osborne Encore **385-387**(2)
Orborne Executive 385(2)
Oscilador 357(2)

P

Page, Jimmy 339(2)
Paleta eletrônica 177-179(1)
Papert, Seymour 31(1)
Partitura eletrônica 337(2)
PASCAL 7, **180-183**, **211-215**(1) 342-344, 372-374(2)
cálculos aritméticos **220-224**(1)
conjuntos operadores **251-253**(1)
estruturação de programas **312-314**(1)
instruções 213-215(1)
matrizes 288-290(1)
organização de arquivos 320-324(2)
uso de funções **320-324**(2)
PC-2001 436(2)
PDSG (Programmable Digital Sound Generator) 398-399(2)
Peddle, Chuck 175-176(1)
Perceptron 257-259(1)
Periféricos **457-461**(2)
PIA (Peripheral Interface Adaptor) 362-364(2)
Piaget, Jean 31(1)
PIPS (Pan Information Processing System) 375(2)
Pirataria **57-58**(1)
Pitts, Walter 257(1)
Pixels 151(1)
PE/1 (Programming Language/1) 7(1), 594(2)
Planilha eletrônica 9, **157-159**, 163(1), 348-349, 355, 390-391(2)
Plotters 60, **282-284**(1), **365-367**(2)
Poesia aleatória **132-134**(1)
Poly 101 HS 356(2)
Pompeu, Renato 552(2)
Porsche, Ferdinand 558(2)
Portas lógicas 28-29, 208, **272-273**, 292(1)
Potter, David 396(2)
Printer-plotters
ver Plotters
Procura lógica 200(1), 459(2)
Programas 5-6(1)
acesso aos **254-255**(1)
aplicativos 291(1)
biblioteca de 275(1)
comerciais 448-449(2)
documentação dos **54-55**(1)
estruturação de **34-35**(1)
geradores de **580-582**(2)
integração de **162-163**, **188-189**(1)
métodos de otimização **294-295**(1)
pacote financeiro **523-524**(2)
projeções em perspectiva **270-271**(1)
rotinas auxiliares **234-235**(1)
testes de **308-309**(1)
Programação 8, **34-35**(1)
erros de **173-174**(1)
módulos 130-131, 144-145(1)
operações lógicas **108-109**(1)
Projeto auxiliado por computador
ver CAD
PROLOG (Programming in Logic) 7(1), 594-595(2)
PROM (Programmable Read Only Memory) 20(1)
Psiquiatra eletrônico **204-205**(1)
PSR (Processor Status Register) 472(2)

Q

Quadrado mágico **44-45**(1)

R

Radiano 340(2)
RAM (Randon Acess Memory) 20(1)
Rascunho eletrônico 553-555(2)
RAT (Remote Action Transmitter) **546-547**(2)
Reatron 336(2)
Reconhecimento visual **589-591**(2)
Recorrência 70, 73, 79, 132-134(1)
Redes de comunicação **37-39**(1) **430-432**(2)
rede local 431-432(2)
rede nacional 432(2)
Registros de controle 362(2)
Resistores 245(1)
RGP (Reação Galvânica da Pele) 478(2)
Rifle óptico **100-101**(1)
Robôs **237-239**(1), **568-571**(2)
braços dos 491-492(2)
de superfície 318-319(2)

do futuro **497-499**(2)
domésticos **317-319**, 577(2)
educacionais **568-571**(2)
fala dos **537-539**(2)
kits de montagem **577-579**(2)
movimentação dos **422-424, 490-492**(2)
sensores 442-444(2)
tartaruga 520-522(2)
visão dos **462-464**(2)
Robótica 317(2)
ROM 20, **128-129**(1), 350-352(2)
RTS (Request To Send) 363(2)

S

Samuel, Arthur 541(2)
Sanyo MBC-550 **465-467**(2)
SB-80 376(2)
Schaeffer, Pierre 338(2)
Seisakujo, Kashio 156(1)
Semicondutor complementar de óxido metálico
ver CMOS
Sensores 443-444(2)
Seqüenciador 358, 437(2)
analógico 382(2)
digital 359(2)
Shannon, Claude 217(1)
Sharp PC-1251 **66-67**(1)
Sharp PC-1500A **66-67**(1)
Sheun Ming Ling 356(2)
Shiina, Takayoshi 375, 556(2)
Shima, Masatoshi 56(1)
Shirahata, Lucas Kenichi 438(2)
Simon, Herbert 258(1)
Simulação 429(2)
de vôo **164-165**(1)
jogos de **428-429**(2)
Sinal
analógico 199(1)
digital 199(1)
de televisão 301-302(1)
Sintesista 438(2)
Sintetizadores 337-339, 418, 437(2)
conceituais 398(2)
de voz 6, 10, 15(1), 441(2)
digitais 359(2)
Sinuca 252(1)
Sistema bancário 76(1)
Sistema numérico 111-112(1)
binário 49(1), 413(2)
hexadecimal 414(2)
Sistema operacional **228-229**, 250(1), **350-352**(2)
de disco
ver DOS
Sistemas especialistas **258-259**(1), **563-565**(2)
Slim drives 14(1)
Snap **246-247**(1)
Software **248-250**(1), 396(2)
formas físicas de
armazenamento 457-461(2)
cartucho 20(1)
disco flexível 14, 21(1), 457-458(2)
fita cassete 20-21(1)
gravador cassete 14(1), 457(2)
slim drives 14(1)
wafadrive **225-227**(1)
Soldagem 68-69(1)
Som 379(2)
Spectravídeo 318 **505-507**(2)
Sprite 108(1)
Starset 138(1)
Stockhausen, Karlheinz 338(2)
String 62, 288-290(1)
Supercomputador **488-489**(2)
Superdotados 139(1)
Synclavier 437(2)

T

Tabela
de decisão 115(1)
de validação 28-29(1)
Tablete gráfico 15, **177-179, 305-307**(1) **402-403**(2)
Tandy Model 100 **586-588**(2)
Tartaruga 16, 30, 32, 50-53(1), 319, 365, **520-522**(2)
Teclado 477(2)
conceitual 138(1)
Tela digital 229(1)
Telecomunicações **197-199**(1)
Televisor 301-302(1)
Terminais 198(1), 460(2)
Testes 308-309(1)
Texto
arquivos de 324(2)
processamento de **17-19**, 137, 226(1), **368-369**, 420-421(2)
TI99/4A 36(1)
TK 90X **186-187**, 276(1)
TK! Solver **348-349**, 390-391(2)
Torode, John 56(1)
Traçador digital **263-265**(1)
Tramiel, Jack 175-176(1)
Transdutores 444(2)
Transistores 245(1)
Trigonometria **242-243**(1), 340-341(2)
Trilhas espirais 151-153(1)
TRS-80 11(1)

U

Unidade Aritmética e Lógica
ver ALU
Unidade Central de Processamento
ver CPU
Unidade de disco 10(1)
Unilamp 336(2)
Unitron **336**(2)

V

Vaucanson, Jacques de 237(1)
Venn, diagramas de 92(1)
Videoclip 359(2)
Vídeo
composto 301-302(1)
interativo 153(1)
monitor de 16(1)
Videogames 10, 116, 276(1)
Videotexto 76(1), 565(2)
projeto Cirandão **548-552**(2)
Visawrite 469(2)
Visão
humana 462(2)
robótica **462-464**(2)
Visicalc 9(1), 355(2)
Visor de cristal líquido
ver LCD
Vôo
simulação de 164-165(1)
Voz
sintetizador de 6, 10, 15(1), 441(2)

W

Wafadrive **225-227**(1)
WANDAH (Writing Aid and Author's Helper) 19(1)
Weir, Sylvia 138-139(1)
Wiener, Norbert 257(1)
Wirth, Niklaus 180(1), 372(2)
World Star 368-369(2)
Wren Executive **425-427**(2)

X

Xadrez **217-219**(1), **540-543**(2)

Y

YAMAHA CX5M **445-447**(2)
Yoshiyuki, Célio 236(1)

Z

Z80 56(1)
Z8000 56(1)

SUMÁRIO

Apresentação 5-8

APPLE
- Ataque Aéreo 9-11
- Golfe Miniatura 17-20
- Letras Mágicas 25-27
- Bloqueie 27-28
- Conecte Cinco 29-31
- Vôo Noturno 37-40
- Palácio de Cristal 45-47
- Aprendendo Matemática 47-48
- Vogal ou Consoante 48
- Zumbi 53-55
- Rastreador 55-56
- Papagaio 60
- Tudo Zero 61-62
- Monstros Galácticos 74-75
- Guerra Galáctica 77-79
- Explosão 79-80
- Pingüim 85-87
- Forças Espaciais (Space Invaders) 99-100
- Pontos de Energia (Archon) 101-102
- Magia Negra (Necromancer) 103-104
- Escolta Aérea (Flyerfox) 105-106
- Fortaleza Espacial (Zaxxon) 107-108
- Um-contra-Um (One on One) 111-112
- Competições Esportivas (Summergames) 117-118
- Sete na Caçapa (Snooker) 119-120

ATARI
- Forças Espaciais (Space Invaders) 99-100
- Pontos de Energia (Archon) 101-102
- Magia Negra (Necromancer) 103-104
- Escolta Aérea (Flyerfox) 105-106
- Fortaleza Espacial (Zaxxon) 107-108
- Caça ao Código Secreto (Impossible Mission) 109-110

Um-contra-Um (One on One) 111-112
- O Inimaginável (Raid Over Moscow) 113-114
- Águas Turbulentas (River Rescue) 115-116
- Competições Esportivas (Summergames) 117-118
- Sete na Caçapa (Snooker) 119-120

COMMODORE 64
- Forças Espaciais (Space Invaders) 99-100
- Pontos de Energia (Archon) 101-102
- Magia Negra (Necromancer) 103-104
- Escolta Aérea (Flyerfox) 105-106
- Fortaleza Espacial (Zaxxon) 107-108
- Caça ao Código Secreto (Impossible Mission) 109-110
- Um-contra-Um (One on One) 111-112
- O Inimaginável (Raid Over Moscow) 113-114
- Águas Turbulentas (River Rescue) 115-116
- Competições Esportivas (Summergames) 117-118
- Sete na Caçapa (Snooker) 119-120

CP-400
- Batalha no Mar 23-24
- PacMan 31-32

MSX
- Nautilus III, A Missão 95-96

SINCLAIR
- Invasor do Espaço 11-12
- Come-Come 15-16
- Teste seu QI 21-22
- Sete e Meio 41-42
- 110 42-43
- Senha 44
- Ordem 62-63
- Ladrão Espacial 63-64
- Indianápolis 73-74
- Vegas 81-82
- Crash 88
- Flagship 93-94
- Invasores 97-98
- Forças Espaciais (Space Invaders) 99-100
- Fortaleza Espacial (Zaxxon) 107-108
- Águas Turbulentas (River Rescue) 115-116

TRS-80
- CRAM 13-15
- Dispare 69-71
- Forças Espaciais (Space Invaders) 99-100

TK 85
- Guerra nas Estrelas 75-76
- Ataque Submarino 83-84

TK 90X
- S.O.S. Atlantis 57-59
- Base Marciana 71-72

VIC-20
- Águas Turbulentas (River Rescue) 115-116

MICROCOMPUTADOR

JOGOS

rioGráfica

CONSELHO DE ADMINISTRAÇÃO
Roberto Irineu Marinho
João Roberto Marinho
José Roberto Marinho
Oscar Neves
Danilo Esteves Costa

Diretoria
Oscar Neves
Danilo Esteves Costa
João J. Noro
Marcos March
Nilo Sérgio de Almeida

GRUPO EDUCAÇÃO E CULTURA

Divisão Comercial

Diretor
Jaime Rodrigues
Gerente de Planejamento
João de Azeredo Coutinho
Gerente de Vendas
Rubens Barbosa

Editorial

Diretor
João J. Noro
Editor-chefe
Maurício Rittner
Diretor de Arte
Adalberto Cornavaca
Administração Editorial
Heitor de Souza Paixão

Editor
Jesse Navarro
Editor-assistente
Cláudio Marcondes
Coordenadora de Revisão
Isis Augusta Loyolla
Revisores
Cacilda Guerra
Carlos Eduardo Russo Tranjan
Luiz Vicente Vieira Filho
Maria da Graça Mendonça Couto
Sérgio Roberto Torres
Chefe de Arte
Bonifácio Duardes Miranda
Chefe de Estúdio
José Yuji Kuribayashi
Assistentes de Arte
Miguel Luis Escámez Simón
Edemilson R. da Silva
Wagner Celestino

Colaboradores
Editor-assistente:
Luiz Carlos Pizarro Marin

Elaboração dos jogos:
Gustavo José Ferreira Grubba
Luiz Henrique Alayon
Renato Oliveira Silva

Fotos:
Fernando Scavone

© APSIF Copenhagen, 1984
© Orbis Publishing Ltd., 1984
© Editora Rio Gráfica Ltda., 1985
para a língua portuguesa

Composição:
Editora Rio Gráfica Ltda.
e AM Produções Gráficas Ltda.

MICROCOMPUTADOR JOGOS

rioGráfica